# CONSTRUCTION AND ADAPTATION OF AI BEHAVIORS IN COMPUTER GAMES

A Thesis
Presented to
The Academic Faculty

by

Manish Mehta

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Interactive Computing

Georgia Institute of Technology
December 2011

# CONSTRUCTION AND ADAPTATION OF AI BEHAVIORS IN COMPUTER GAMES

Approved by:

Ashwin Ram, Advisor
School of Interactive Computing
*Georgia Institute of Technology*

Mark Riedl
School of Interactive Computing
*Georgia Institute of Technology*

Blair MacIntyre
School of Interactive Computing
*Georgia Institute of Technology*

Andrew Gordon
Institute for Creative Technologies
*University of Southern California*

Brian Magerko
School of Literature, Communication and Culture
*Georgia Institute of Technology*

Date Approved: 15 Aug 2011

*To my beloved wife and my parents,*

*who beared my long hours working on the proposal and who showed immense patience in helping me get through this difficult journey.*

# ACKNOWLEDGEMENTS

First of all I would like to thank my family. Their love and support has been invaluable. My wife who has always believed in me, helped keep life in perspective and supported me through tough days when things were not working as I would have liked them to. I would like to thank my parents. They have encouraged me to follow my dreams and have supported me in all the decisions I have taken in my life.

I am especially grateful to my advisor, Dr. Ashwin Ram for not only helping me develop as a researcher but as an individual as well. He has played a fundamental role in the formulation of the topic of this thesis and helped shape this research.

I want to thank my committee members Dr. Brian Magerko, Dr. Mark Riedl, Dr. Blair MacIntyre and Dr. Andrew Gordon for their insights, posing thoughtful questions, helping me with new ideas and their overall inputs throughout the process.

I would also like to thank members of the Cognitive Computing Lab- Dr. Santi Ontanon and Saurav Sahay for their support and encouragement throughout the process. A special thanks to Brian Sherwell, Gabriel Cebrian and Paritosh Mohan who have helped in developing the thesis system. I would like to acknowledge the contribution of Dr. Andrea Corradini- without his help and support it would not have been possible to finish the evaluation of the dissertation system and analysis of the collected data. I want to acknowledge the support of Dr. Michael Mateas for his encouragement during the initial years of my PhD career that helped formulate some of the research directions undertaken in this thesis.

The thesis work has been long and has required support of many individuals along the way. I would like to acknowledge here a few of those individuals who although not directly quoted in the text have helped push this work forward. I apologize if I

may have missed names of other individuals who have contributed. You all deserve a very special thanks.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# SUMMARY

Computer games are an increasingly popular application for Artificial Intelligence (AI) research, and conversely AI is an increasingly popular selling point for commercial digital games. AI for non playing characters (NPC) in computer games tends to come from people with computing skills well beyond the average user. The prime reason behind the lack of involvement of novice users in creating AI behaviors for NPC's in computer games is that construction of high quality AI behaviors is a hard problem. There are two reasons for it. First, creating a set of AI behavior requires specialized skills in design and programming. The nature of the process restricts it to certain individuals who have a certain expertise in this area. There is little understanding of how the behavior authoring process can be simplified with easy-to-use authoring environments so that novice users (without programming and design experience) can carry out the behavior authoring task. Second, the constructed AI behaviors have problems and bugs in them which cause a break in player experience when the problematic behaviors repeatedly fail. It is harder for novice users to identify, modify and correct problems with the authored behavior sets as they do not have the necessary debugging and design experience. The two issues give rise to a couple of interesting questions that need to be investigated: a) How can the AI behavior construction process be simplified so that a novice user (without programming and design experience) can easily conduct the authoring activity and b) How can the novice users be supported to help them identify and correct problems with the authored behavior sets?

In this thesis, I explore the issues related to the problems highlighted and propose a solution to them within an application domain, named Second Mind(SM). In SM

novice users who do not have expertise in computer programming employ an authoring interface to design behaviors for intelligent virtual characters performing a service in a virtual world. These services range from shopkeepers to museum hosts. The constructed behaviors are further repaired using an AI based approach. To evaluate the construction and repair approach, we conduct experiments with human subjects.

Based on developing and evaluating the solution, I claim that a design solution with behavior timeline based interaction design approach for behavior construction supported by an understandable vocabulary and reduced feature representation formalism enables novice users to author AI behaviors in an easy and understandable manner for NPCs performing a service in a virtual world. I further claim that an introspective reasoning approach based on comparison of successful and unsuccessful execution traces can be used as a means to successfully identify breaks in player experience and modify the failures to improve the experience of the player interacting with NPCs performing a service in a virtual world. The work contributes in the following three ways by providing: 1) a novel introspective reasoning approach for successfully detecting and repairing failures in AI behaviors for NPCs performing a service in a virtual world.; 2) a novice user understandable authoring environment to help them create AI behaviors for NPCs performing a service in a virtual world in an easy and understandable manner; and 3) Design, debugging and testing scaffolding to help novice users modify their authored AI behaviors and achieve higher quality modified AI behaviors compared to their original unmodified behaviors.

# CHAPTER I

# INTRODUCTION

The design of interactive gaming experiences is increasingly important in our society. Examples include interactive media, video games, and interactive portals where users interact with a computing system in a natural and engaging manner. There is increasing interest in modes of interaction with virtual characters, as it represents a more natural way for humans to interact with computing systems. Unlike traditional design (which focuses on design of artifacts) and traditional human-computer interaction (which focuses on design of interfaces), natural interaction with virtual characters requires also the design of personalities and behaviors of so-called "believable characters".

Creating believable characters is a complex task, typically[1] requiring both creative skills (to design characters with personalities, emotions, expressions, gestures, responses, behaviors) and programming skills (to code these in a low-level scripting or programming language). Modeling systems (such as 3D Studio Max or Maya) enable designers to create character models and animations, but developing and coding behavior scripts is still very much a black art. There is little understanding of how the behavior authoring process can be simplified with easy-to-use authoring environments so that novice users (without programming and design experience) can carry out the creative task. Debugging issues with the constructed AI behavior sets presents another problem. It is harder for novice users who do not have the debugging and design experience to identify and correct problems with the authored behavior

[1] There are techniques like learning from demonstration that allow creating interactive characters and do not require programming skills

sets and improve them. Repeated failures in the authored behaviors can potentially cause a break in player experience. Thus, initial construction of AI behavior by novice users and their subsequent repair presents two key challenges that need to be addressed in order to allow novice users to create high quality AI behaviors. Based on these challenges, I have identified two problems that also represent my thesis problem statement.. These problems are as follows:

I. *How do we provide novice users (without programming and design experience) the necessary support to easily author AI behaviors?*

II. *What kind of advanced AI approaches can help novice users to identify and repair problems with the authored AI behaviors?*

I provide solution to these problems though an authoring environment (named Second Mind) where novice users who do not have expertise in computer programming and design skills can easily author behaviors for virtual characters. These behaviors are essentially sequence of basic actions (like talk, gestures etc) that can be triggered in response to the game world state. The behaviors are created for NPCs performing a service in a virtual world. These services range from shopkeeper to museum hosts. An AI based approach in the authoring environment supports the authors in identifying and repairing problems with their authored behavior sets and helps them improve the quality of behaviors. The repaired behavior sets provides a better experience for players who interact with them compared to the original unrepaired behavior sets.

To evaluate the solution I conducted a four-part evaluation study with human subjects. The evaluation results indicate that the behavior authoring and repair solution provide authors with a) a programming scaffolding through easy to use authoring environment, understandable vocabulary and integration with a virtual world to construct the behaviors, b) a testing scaffolding to receive feedback from the players, c)

a debugging scaffolding to help the authors identify problems with the authored behavior sets and d) a design scaffolding by suggesting them new behaviors they could add to their existing behavior sets. The design, debugging and testing scaffolding improves the quality of the authored behavior sets since it achieves a better subjective experience for players who interact with the authored behavior sets.

The rest of the chapter is organized as follows. In the next section, I discuss the challenges computer games pose for AI approaches situated within them. In section 1.2, I delve a little deeper into the problem that this thesis explores through two illustrative example scenarios from a computer game and presents the research questions that have been formulated to address the problem. I further discuss the necessity of addressing the problems that this thesis explores in section 1.5. Section 1.6 revisits the research questions posed to address the problems along with their answers. I also discuss the claims and present the thesis statement. I present thesis solution and evaluation in section 1.7 and contributions to the research community in section 1.8. In the end, I present an overview of the dissertation in section 1.9.

## *1.1 Challenges in Computer Game AI*

Computer games pose interesting challenges for artificial intelligence community. Some of these challenges are the following:

(a) *Complex decision spaces*: Most state-of-the-art computer games involve complex strategic (real-time strategy games) or believable behaviors (interactive stories). Both kind of behaviors share the characteristic of having huge decision spaces, and thus traditional search-based AI techniques cannot be applied. Learning techniques or higher level representations are required to deal with such complex games. Traditionally, computer games use handcrafted strategies coded by the game developers, but these tend to be repetitive, and players easily find holes and exploit them.

(b) *Knowledge engineering*: Even assuming that strategies or behaviors are hand-crafted, authoring these behavior sets in a game requires a huge human engineering effort. Game developers have to encode all the knowledge they have about a domain (either to achieve a strategic behavior or a believable human behavior) in some sort of behavior language.

(c) *Authoring support for novices*: Hand crafted behaviors are, ultimately, software code in a complex programming language, prone to human errors. The behavior errors could be in the form of program 'bugs' or not achieving the desired result. Tools are needed to support authors, who are typically not artificial intelligence experts, to author behaviors in a computer programming language.

(d) *End user support*: End users are involved in creating variations of the released computer games (called game mods) with the help of tools released along with them. Although user-generated game content has been around for years, successful game mods tend to come from people with graphics and computing skills well beyond the average user. But as game technology has become more sophisticated, fewer and fewer amateurs/novices have been able to keep up. Thus AI authoring by end users in game mods is very limited due to the special expertise in development and programming experience required for it.

(e) *Unanticipated situations*: Typically it is not possible to anticipate all possible situations and player strategies that can be encountered during game play. This makes it difficult to craft behaviors that react in an appropriate manner to these unforeseen circumstances and player actions.

(f) *Human in the loop*: State of the art games involve an interactive user that can change the game state instantaneously. Any considerable delay in AI decision making could result in a decision impertinent to the current game situation.

(g) *Real time nature*: Interactive games operate in real time. Their real time nature impose constraints in terms of processing time and fewer trials that could be taken by AI approaches situated within these domains.

(h) *Believable character performance*: In interactive games, AI agents typically have their own personalities, affecting the way they act in the game. Characters need to autonomously exhibit their author-specified personalities in new and unforeseen circumstances. Non-believable responses of game characters in role-playing or adventure games breaks the illusion of an ongoing narrative, exposing limitation of built-in AI.

(i) *User-specific adaptation*: Different players may enjoy different strategies to fight against (in the case of real-time strategy games), or different styles of story telling (in the case of interactive dramas), different types of story development, different kinds of character behaviors and interactions, and/or different educational problems. As game designers begin to include user modeling capabilities, the AI strategy and behavior must, in turn, be adaptable based on the user model.

(j) *Replayability and variability*: A player might get bored of seeing the same strategies and behaviors over and over again. Although simple variability can be achieved through stochastic selection of behaviors or strategies from a large repository, this increases the authoring burden. Furthermore, random selection begs the question of true interestingness.

This list is not exhaustive, but is intended to give a flavor of the kind of problems that computer games pose to the AI community. A conclusion that can be drawn from the previous list is that not only can games benefit from better AI techniques, but AI can also benefit from the challenges that computer games provide.

This thesis uses interactive games as domain of research looking at the different challenges (c-g). The problem explored in these interactive domains is instead the one

**Figure 1:** A snapshot of a commercial interactive story ("And then there were none")

stressed in point i) **C**onstruction of AI behaviors requires specialized skills (looking at the challenge as expressed in item d) and ii) Problems with the created AI behavior sets results in break in player experience when the behaviors repeatedly fail. (looking at the challenges c and e-g).

## *1.2 The Problem and Research Questions*

Let us discuss two illustrative example scenarios from a computer game to understand the problem that this thesis explores.

### 1.2.1 Illustrative Examples

#### *1.2.1.1 Scenario 1*

Consider that Steven (a game player) buys an interactive story game that has been recently released based on the story "And then there were none". The screenshot of an actual commercial game is shown in the Figure 1. The story context is that the player has been invited to have dinner to a manor. When the dinner is about to start, the player discovers that one of the other guests has been murdered. After the introduction, the player is free to move around and interact with the other characters in the house. The goal is to find out who among the guests in the house was the assassin. After having played the game, Steven wants to show off his creative skills

by adding a few more simplistic scenes to the game to add more variability for other online players. Steven might have a few simplistic scenes in mind that involve only a few characters with a particular personality, emotions, and background. Authoring such experience is an example of the creative process that I wish to understand and enhance. Using the traditional approach (i.e., hand-authoring scripts) would require a tremendous effort in authoring each one of the individual characters. Moreover, as Steven doesn't have the programming experience, he cannot add new scenes to the game. If there was an environment that allowed more people like Steven who are non-programmers and essentially end game users to be involved in extending the game, it would provide them an opportunity to exercise their creative skills and make them available for other players. The game play experience would now improve as issues with replayability of static game could be addressed to some extent given a large base of users creating content and sharing it with each other. Typically game companies only provide toolkits to create scenes/maps. The behavior creation is carried out through scripts in a programming language which restricts the process to people with a certain degree of programming skills.

#### *1.2.1.2 Scenario 2*

Let us consider another example from the same game. In the second scenario, we assume that a novice author has created an initial set of behaviors. Creating behaviors for the characters required an enormous amount of work to properly author the behavior of all the characters so that they can properly react to their environment (this includes the player). However, as it is not possible to envision all the possible circumstances that the characters would encounter in the game, there would be circumstances where a character would start behaving differently from his intended personality. As the novice author would have difficulty identifying the problems in the constructed behavior sets, the characters would continue to execute the failed

behaviors (that doesn't meet his personality style). If there was a repair layer added to the characters AI system, the failures in the behavior sets could be identified and repaired appropriately to match the changing context and personality. The authors could be presented with the possible repairs so that they can see the problems with the created behaviors and add/modify things based on them.

These illustrative examples highlight the kind of problems this dissertation addresses. Scenario 1 highlights the first aspect, i.e. the problem of addressing "construction of AI behaviors by novice users" that this thesis will explore. Scenario 2 highlights the second aspect i.e. the problem of "repair of novice authored AI behaviors". At this point it should be noted that the term novice users is used broadly to define the user population that do no have any skills and experience in both programming and design.

## *1.3 Understanding the problem of AI behavior construction by novice users*

Novice users bring little or no programming experience to the behavior construction task. However, users have a combination of real-world experiences, experience with other software, and with computers in general that they will employ to understand the construction task. The fundamental part of human cognition is to try to relate something new that they encounter in daily lives to something they already know. Familiar interaction design approaches that allow users to create correspondences with a more familiar domain [47] have been used in product design and have been shown to make a product or interface easier to understand. The key question, therefore boils down to identifying the right interaction design approach that can be employed as part of presentation of the authoring process that can help the novice users understand the construction process.

The AI behavior construction process requires an underlying representation formalism and vocabulary that can be used to represent the behaviors. Representations

have been a keenly studied topic within the field of artificial intelligence. Many representational formalism have been proposed in AI, ranging from predicate calculus through frames and scripts to semantic networks and more. An AI scientist, who is familiar with the vocabulary of the underlying representational formalism is adept at translation between the domain that is being represented and the formal language used as the representation. A novice user, however, would not have the necessary background to use the formal representational language. One of the criteria along which different AI representation formalisms are classified is their expressivity. The more expressive, the easier it is to represent something. The key requirement for a representation formalism and the corresponding vocabulary designed for novice user however, is understandability. Some level of expressivity can be forsaken for better understandability. The representation vocabulary should speak the user's language and use words, phrases, and concepts familiar to the user, rather than terms more familiar to an AI scientist.

Summarizing, a design solution that will enable novice authors to construct AI behaviors should provide the right interaction design approach and representation vocabulary to support them in the authoring activity. The first research questions therefore reads as follows:

- **Research Question 1: What is a design solution that enables novice users to easily create AI behaviors for NPCs performing a service in a virtual world?**

For tractability, we focus on the construction of particular kinds of AI behaviors situated in a particular domain. The scope of AI behavior creation is confined to creation of action sequences for NPCs performing a service in a virtual world. The behaviors can be activated based on author defined activation triggers.

## *1.4 Understanding the Problem in repairing novice authored AI behaviors*

Identifying and repairing the failures in the constructed AI behaviors set presents another problem. There are several dimensions to this problem which make it an especially hard and interesting problem. In general,

- there are several metrics along which the constructed AI behaviors can be improved and the behaviors could be appropriately modified to improve things along these.
- there are several stages at which the need for repair of the AI behaviors may be detected.
- there are several agents who could provide the feedback on whether the AI behavior performance is according to the desired performance metric.
- there are several types of feedback that can be provided to repair the AI behaviors.
- there are several stages during which the repair could be carried out.
- the responsibility of repair could be divided between the AI system and the system designer.

Lets look at each of these dimensions in turn.

**Performance metrics**: Different game genres present various performance requirements for a repair approach situated within them. For instance, god games usually require the game AI to solve resource allocation problems and solve long-term strategy problems, while interactive drama require the game AI to adapt the story according to the player interactions in a way that it is more appealing to the player (thus, the latter requires user modeling and story planning). Moreover, adventures, interactive dramas and other genres with embodied characters usually require

believable behaviors. In principle, there may be several different properties desired of the repair approach. Let us look at a few of the performance metrics desired of repair AI approach:

- *Believable Character Behavior*: the goal of the AI based repair approach for these characters is to behave in accordance with their personality.
- *Strong AI Performance*: the goal of the resulting AI based repair approach is to achieve a more winning percentage and thus provide a more challenging opponent to the player.
- *Better Player Experience*: the goal of the resulting AI based approach is to provide the player into a better playing experience.
- *Adaptation to Challenge Level*: the goal of the AI based repair approach is to continuously scale a game's difficulty level to the point that the human player is challenged, but not completely overpowered. As a result, the game will be most entertaining and will remain entertaining even if player's skill level increases.

These four different metrics are not mutually exclusive. Any AI based repair approach that provides for a stronger AI could possibly provide for a better player experience as argued by other researchers [18]. Similarly a believable performance by the characters could possibly result in a more pleasurable experience for the player.

**Performance measurement stage**: The performance of the resulting AI behaviors could be measured at two different points in the game episode.

- either, during the performance of the AI behavior sets in the game episode itself.
- or, after the completion of the game episode.

**Feedback agent**: The feedback on these metrics could be provided in different ways. The game environment itself could provide the feedback that the performance

was not according to the desired criteria (on the various performance metrics defined above). The designer of the system or the player interacting with the system could also provide the feedback.

**Type of feedback**: The type of feedback could be different. For example, the feedback may simply inform the system whether the solution it produced succeeded or failed. In some cases, the feedback may consist of a complete trace of the execution of the solution produced by the system, localizing the steps which need modification. In others, the designer himself may provide another preferred solution along with a complete trace of how this solution could be produced. Alternatively, the feedback may only specify the alternative preferred solution.

**Repair of AI**: The repair can be carried out at different stages. It could be during the performance of the AI behaviors itself or after the completion of problem solving.

**Responsibility of modification**: The responsibility of repair could lie solely with an AI system i.e. fully autonomous repair (without any designer intervention) or could be shared between the system and the designer (novice users in our case) in some way carrying out a mixed initiative repair.

As we discussed earlier in section 1.3, in this thesis, we have used player experience as a way of measuring the quality of the authored AI behavior sets. Player feedback provides a way to measure if the AI behavior sets are meeting the performance quality desired of them. Based on player experience, if the need for repair of AI behaviors is detected, the AI repair solution should then identify problems with the constructed behavior sets. The second research question therefore reads as follows:

- **Research Question 2: How can a system automatically identify failures in human authored AI behaviors for NPCs performing a service in a virtual world?**

Once the failures and problems are identified, the actual repair needs to be carried out so that the repaired behavior sets could improve the player experience. The third research question therefore reads as follows:

- **Research Question 3: How can a system repair failures in human authored AI behaviors to improve player experience?**

In the next section, I analyze the need for addressing the problem looked at in this thesis. I start off the section by taking a step back by first looking at the broader question of what is the need for tools for novice users. The first question has been explored by researchers in other fields who have addressed the merit in developing tools for novice users to carry out part of product development and programming activities. I argue that this broader perspective also applies to the development of environment that simplifies the AI behavior authoring process. I look into their perspective to get some insight into the need for AI behavior construction tools for novice users.

## *1.5 Purpose of Research*

### 1.5.1 Why is the question of creating tools for Novice Users important?

There are a series of reasons that makes the AI creation tools for novice users an important issue.

- **End user innovation improves products**: End User appropriation is a commonly used term to describe ways in which customers engage with the product, using, changing and adapting them to transform its original design. Appropriating a product has been described as a common end user activity [83], often carried out in ways unforeseen by the original designers [32]. As a result, end user appropriation is also referred to as second stage of design [19]. Eric von Hippel has convincingly argued that users participation in research and

development, e.g. through user appropriation, can contribute substantially to innovation and product improvement [111, 21]. Empirical studies on the sources of innovation have also revealed that in the fields of both industrial and consumer goods, users are often the initial developers of products, prototypes and processes which later gain commercial significance [89]. Extending the same argument, I believe that opening up the tools of AI production for games, education, home robots and other AI artifacts would allow for better and creative products for these domains as well.

- **Broad range of people get involved**: Having people from a broad range of areas involved allows a medium to grow and drives the production of newer ideas. Restricting a medium to a few set of individual's results in "homogeneity of expression" [23]. A similar argument has been raised for simplifying programming languages for novice programmers. Kelleher and Pausch argue that novice programming environments would attract more diverse group of people to computer science [57]. In a similar vein, if the AI behavior authoring process is restricted to people who are essentially programmers, the process would be restricted to a self-selected group with a relatively homogeneous set of skills, interests, experiences and backgrounds. Mateas argues that AI opens up new opportunities in art and entertainment, enabling rich and deeply interactive experiences [69]. Opening up the tools for AI creation for novice users will have the same effect and provide novice users an avenue to exercise their creative freedom allowing possibilities for newer kind of AI artifacts.

- **User needs are better captured**: User involvement in the development of new products can offer a novel approach to improved methods of meeting customer needs [111]. Hippel points out that the primary function of toolkits for user design is to co-locate product development and service-development tasks

with the information needed to execute them. Need-intensive tasks assigned to users, along with the tools needed to carry those tasks out. At the same time, solution-intensive tasks are assigned to manufacturers. A similar argument has been raised by Kelleher and Pausch with the goal towards simplifying programming languages and environments for novice users [57]. A more even match between population of people developing software to the population using it would provide a better match users' needs. Design of AI products that match more closely to the user needs would be produced as a result of simplification of AI creation tools. People who are using the AI artifacts would be able to easily modify the product to suit their needs.

We see from the above list that there is merit in simplifying tools so that end users can be involved. Having end user involved in AI creation allows for posing and answering novel research questions that would not be raised unless doing AI research in this context. This thesis is driven by the belief that AI behavior authoring should be brought to the novice user. This cannot happen unless the technology is made easier and more accessible to people outside the current community of (very few) AI programmers. Current barriers to the technology are simply too high.

### 1.5.2 Why repair novice authored AI behaviors?

The following are some of the reasons on why it is important to repair novice authored AI behaviors:.

- **Better player experience**: due to failures in the authored behavior sets, a player might get bored of seeing the problems in the behaviors again and again. Different players may enjoy different strategies to fight against (in the case of real time strategy games), or different styles of story telling (in the case of interactive dramas), different types of story development, different kinds of character behaviors and interactions, or different educational problems. Modifying the AI

strategy and behavior based on the player preferences can help provide a better player experience.

- **Better handling of unanticipated situations**: it is not possible to anticipate all possible situations and player strategies that can encountered during game play. This makes it difficult to craft AI behaviors that react in an appropriate manner to these unforeseen circumstances and player actions. Modifying the AI behaviors to these unforeseen circumstances would allow for better handling of the changing circumstances.

- **Achieving game objectives**: it is possible, even likely, that authored behaviors or strategies do not achieve the games objectives adequately, especially in realistic, scaled-up domains or applications. These objectives could range from entertainment to education, training, etc. Thus, the game has to realize that the objectives are not being met on a per-use basis, and modify the AI behaviors accordingly. For example, a particular user may be getting bored, or not learning the intended lesson.

- **Stronger AI**: Many researchers and game developers hold that game AI in principle is entertaining when it is difficult to defeat [18]. Expert gamers in strategy games, quickly get bored of playing against a built-in AI, as they can easily find and exploit flaws in the AI. Modifying the AI behaviors in response to newer player tactics to improve its performance would allow for a more challenging AI .

- **Believable character performance**: Believable agents are autonomous characters exhibiting rich personalities, emotions, and social interactions. Over long game sessions, a character's static behavioral repertoire may result in repetitive behavior. When behaviors fail to achieve their desired purpose, characters are unable to identify such failure and will continue executing them. Such repetition

harms the believability of the characters. Appropriate repair of the AI behavior sets would allow these characters to autonomously exhibit their author-specified personalities in new and unforeseen circumstances.

We see from the above list that there is merit in repairing problems with the AI behaviors. Next, let's look at the research questions, their answers and claims made.

## *1.6 Thesis Statement and Claims*

In Section 1.2, I formulated four research questions to address this problem. These questions are presented again here along with the answers. I also present the claims that I make.

### 1.6.1 Construction with AI Behaviors

One of the key challenges in making the AI behavior construction process simpler is to identify interaction design approaches, languages, and tools that enhance the authoring environment's understandability. This raises the following key research question that need to addressed:

- **Research Question 1: What is a design solution that enables novice users to easily create AI behaviors for NPCs performing a service in a virtual world?**:

Users should be able to infer how the programming system works by referring to their existing knowledge and expectations about how the modeled system works. My answer to this question is the following:

| |
|---|
| A design solution with behavior timeline based interaction design approach for behavior construction supported by an understandable vocabulary and reduced feature representation formalism enables novice users to create AI behaviors in an easy and understandable way for NPCs performing a service in a virtual world |

### 1.6.2 Repairing AI behaviors Research Questions

The second research question is the following:

- **Research Question 2: How can a system automatically identify failures in human authored AI behaviors for NPCs performing a service in a virtual world?**

My answer to this question is the following:

| |
|---|
| Comparison of successful and unsuccessful execution traces can be used to identify failures in human-authored AI behaviors for NPCs performing a service in a virtual world. |

The third research question is the following:

- **Research Question 3: How can a system repair failures in human authored AI behaviors to improve player experience?**

My answer to this question is the following:

| |
|---|
| Execution traces where the identified failures do not exist allow successful repair of identified failures in AI behaviors for NPCs performing a service in a virtual world. The success of repair is measured in terms of improvement in the player experience ratings of the repaired AI behaviors compared to novice users original unrepaired and self-repaired AI behaviors. |

### 1.6.3 Thesis Claims

One the basis of work conducted in this thesis, I make the following claims:

I. **A design solution with a behavior timeline based interaction design approach supported by an understandable vocabulary and reduced feature representation formalism enables novice users to author AI behaviors in an easy and understandable manner. The scope of AI behavior creation is confined to creation of action sequences for NPCs**

**performing a service in a virtual world that can be activated based on author defined activation triggers.**

II. **An introspective reasoning approach based on comparison of successful and unsuccessful execution traces can be used as a means to successfully identify failures in player experience and modify the failures to improve the experience of the player interacting with NPCs performing a service in a virtual world.**

The scope of construction and repair of the AI behaviors is restricted to action sequences for non-playing characters in role-playing scenarios where the roles range from acting as a shopkeeper and museum host.

The claims leads to the following thesis statement:

**An introspective reasoning approach for repairing behavior failures supported by a behavior timeline based behavior construction process, understandable vocabulary and reduced feature representation formalism enables novice users to achieve behavior quality comparable to AI experts. The scope of AI behavior creation and repair is confined to creation of action sequences for NPCs performing a service in a virtual world that can be activated based on author defined activation triggers.**

Next, I will describe the thesis solution, evaluation and contributions.

## *1.7 Thesis Solution and Evaluation*

The authoring solution (named Second Mind) consists of three parts. First, the behavior authoring process is represented as designing agents mind in a graphical environment. The user is presented with basic building blocks that can be combined together on a timeline to construct behavioral entities. Timelines are employed by most popular professional video editing packages and builds on existing metaphors employed by novice users. Timelines indicate a left-to-right sequence, where each

event occurs in chronological order. The behavior construction process represented as a sequence of actions on a timeline in a mind design metaphor makes it easier for novice users to conceptualize the process. Second, through an iterative development approach, a representational formalism and vocabulary for authoring process is created. The vocabulary is made intuitive and understandable for novice users. The complexity and feature set of the representation are reduced but it is still sufficient enough to represent and execute the behaviors. Third, a behavior execution framework is provided that executes the authored behavior in a concrete virtual world environment named Second Life (SL), thus allowing novice designer to observe the simulation of resulting behaviors showing the interaction with other characters.

The work is focused on creating island hosts in SL. One of the current problems in SL is the empty islands when you visit these virtual spaces. The authoring system is targeted towards creating island hosts in SL. These island hosts are created for various different types of virtual spaces in SL and perform different kind of roles. These roles range from shopkeepers selling goods like furnitures, DVD's and clothes or a receptionist for a university department.

Once the behavior sets are authored, they are repaired using an approach consisting of three parts. First during authoring, behaviors and overall interaction are associated with constraints that provide a way to measure the success of the behavior sets and the overall interaction. Second, players (who interact with the authored avatars) provide feedback at the end of their interaction by answering questions on executed behaviors performance and overall interaction. These questions ask for feedback on the performance of the behaviors in relation to the associated behavioral constraints and overall interactions(defined in the first step). These numerical feedbacks help measure the success of the behaviors and overall interaction in relation to the associated behavioral constraints and overall interaction constraints. Third, the repair of behaviors is carried out through comparison of successful and unsuccessful

execution traces. These execution traces are a representation of events happening during the interaction of player with the avatar. Successful execution traces have a high overall interaction rating whereas unsuccessful traces have a low overall interaction rating provided by the player. The comparison of successful and unsuccessful traces provides a way to identify failures in the unsuccessful behavior sets. Once the problems are identified, their repair involves looking at execution traces (where these failures do not exist) and to suggest modifications to the author.

In order to support the claims, a four phase evaluation study has been carried out. In the first phase of the evaluation study, six category of users (UI designers, AI programmers, programmers, Acting background non-programmers, Second Life non-programmers and non-programmers with varied backgrounds are invited to author shopkeeper behaviors for a Second Life island using the authoring system. In the second phase, another set of users (we call them players) interact with the shopkeeper avatars (created as part of the first step) in SL. This provides the necessary feedback for AI repair system to improve the behavior sets. In the third phase, a subset of the same user set who carried out behavior authoring in first phase are invited back. Authors are first presented with text based feedback on how well the behavior sets performed. These text based feedback are simple textual description detailing the performance of the behaviors with respect to associated constraints. The text based feedback is based on feedback provided in the second step. The authors modify the behaviors based on the suggestions. In the second step, authors are presented with the suggested changes from the AI repair approach. Authors take these AI suggestions as the starting point and either accepted them as it is or modify the changes further. In the fourth step, another set of players interact with four sets of behaviors. The first set is the original behaviors created in the first phase. The second set of behaviors are the ones that have been modified based on authors themselves by looking at the text based feedback. The third set of behaviors is the one modified by the AI (without any

author intervention) and the fourth set is a combination of modification by the AI and the authors. Players interact with the four sets of behaviors in Second Life and provide feedback on their interaction. The ratings for all the four sets of behaviors provide a way of measuring AI repair approach effectiveness in appropriately modifying the behavior sets and achieving a better player experience.

Based on the results of the evaluation, we can next discuss the contributions made.

## *1.8 Thesis Contributions*

As contributions to the research community, my dissertation work presents the following contributions, a) Novice user understandable authoring environment to help them author AI behaviors in an easy and understandable manner; b) Design, debugging and testing scaffolding to help novice users achieve higher quality AI behaviors compared to their original unmodified behaviors; and c) a novel introspective reasoning approach for successfully detecting and repairing failures in AI behaviors. The scope of AI behavior creation and repair is confined to creation of action sequences for NPCs performing a service in a virtual world that can be activated based on author defined activation triggers. Let us discuss each of the contributions in more detail.

### 1.8.1 Novice user understandable AI behavior authoring environment for constructing AI behaviors

Creating AI behavior sets require specialized skills in development and coding. There is little existing work on understanding how the behavior authoring process can be simplified with easy-to-use authoring environments so that novice users (without programming and design experience) can carry out the creative task. Creating such a behavior authoring environment requires representation vocabulary and tools that enhance the authoring environment's understandability. The work presented in this thesis provides design, implementation and evaluation of a novice user understandable AI behavior authoring environment that allows novice users to author AI behaviors

in an easy and understandable manner through,

- *Novice user understandable AI vocabulary:* that is used for representing the authoring task.
- *Understandable interaction design approach:* that helps the authors easily create behaviors.
- *Executable framework for behavior evaluation:* that helps the author evaluate constructed AI behaviors.

Some of the existing work on authoring environments is focussed on providing authors the necessary tools to create interactive stories [100, 58, 29, 114, 95, 44, 88] whereas my dissertation work is focussed on proving authoring support for creating AI behaviors for the characters who are part of the story. Other works have used approaches like learning from demonstration to author behaviors for robotic or game agents [82, 81]. However using learning from demonstration as the sole way of authoring behaviors takes away the authorial expressivity that comes when using an authoring environment where the authors could have more fine-grained control over the authoring process.

Second Mind employs a timeline based interaction design approach for presenting the behavior construction process. Timelines are employed by most popular professional video and movie editing packages. Timelines indicate a left-to-right sequential sequence, where each event occurs in chronological order. The construction of behavior in the second mind authoring system involves creating a left-to-right action sequence represented on a timeline. Timelines have been employed as part of authoring systems for creating interactive dramas as well[72]. The key contribution of my thesis work is not the invention of timelines as an interaction design approach for AI behavior authoring but in empirically demonstrating that a) a left to right sequencing (represented by a timeline based interaction design approach) represents a more

natural way for novices to think about the behavior construction process and b) a timeline based interaction design approach can successfully enable novice authors to create AI behaviors in an easy and understandable manner.

In Chapter 3, I present the first design pass over a representational vocabulary and authoring activity that a novice user needs to carry out the authoring of AI behaviors. The evaluation presented in Chapter 3 provides key design insights for the construction of authoring interfaces for AI behaviors. It identifies the problems that the authoring interface designed for novice users needed to address as well. In Chapter 4, I present the current authoring interface and how some of the issues that novice users face can be addressed through improvements in the authoring interface. Results from the evaluation of the authoring interface (presented in Chapter 6) provides evidence that novice users are able to use the current authoring interface to construct AI behaviors that are as good or in some cases better than the experts in the domain. The design and implementation of the authoring interface sheds light on a range of architectural and conceptual issues and solutions to these issues that will be relevant to any future authoring environment intended for creating AI behaviors by novice users.

### 1.8.2 Design, debugging and testing scaffolding for authored behaviors

Once the AI behaviors are authored, they still have problems and bugs. This causes a break in player experience when the behaviors repeatedly fail. It is harder for novice users who do not have the debugging and design experience to identify and successfully correct problems with the authored behavior sets and improve them. The work presented in this thesis provides authoring support in design, debugging and testing of authored behaviors through,

- *Design scaffolding:* for the authors that provided them with new ideas on adding behaviors and thereby creating a richer player interaction.

- *Debugging scaffolding:* that helps the authors to debug the authored behavior sets and identify problems with them.

- *Testing scaffolding:* for authors to assign constraints on the authored behavior sets and gather player feedback on the constructed behaviors.

The design, debugging and testing scaffolding allow authors to identify and successfully correct problems with the authored behavior sets. The success of the corrected behaviors is measured in terms of improvement in the player experience ratings of the corrected AI behaviors compared to human authored original uncorrected and self-corrected (without any scaffolding) AI behaviors.

Existing work in various authoring environments like story authoring environments [100, 58, 29], for example, is focused on providing necessary support for construction activity but does not provide support for identifying problems with the constructed artifacts. My work in this dissertation is unique in this aspect that it also provides the necessary design, debugging and testing scaffolding for authors to identify issues with the authored behavior sets. This becomes especially critical in the case of novice authors who do not have the necessary background to identify problems with the authored behavior sets. In Chapter 4, I present the current authoring interface and discuss the testing scaffolding provided by it. In Chapter 5, I present the approach for design and debugging scaffolding in the authoring interface. Results from the evaluation of the authoring interface presented in Chapter 6 provides support that authoring interface is able to provide authors the necessary support for design, debugging and testing of authored AI behaviors to improve the constructed AI behaviors.

### 1.8.3 Novel introspective reasoning approach

In order to provide the design and debugging scaffolding requires advanced AI algorithms for identifying failures and conducting repairs on AI behavior sets. I have developed AI algorithms and techniques based on introspective reasoning for failure

identification and performing corresponding repair on the authored behavior sets. The repair approach addresses the following three key issues,

- *Detecting need for repair:* The repair approach work presents an approach for detecting the need for repair through declarative behavior constraints assigned by the authors and player feedback that provides feedback on the constraints after the interaction with the avatar.

- *Failure Identification:* The AI algorithms provide a novel approach for identifying the failures through comparison of successful and unsuccessful execution traces.

- *Repair Methodology:* The repair approach provides a novel approach of repairing the failures using execution traces where these failures do not exist.

The success of repair is measured in terms of improvement in the player experience ratings of the repaired AI behaviors compared to human authored original unrepaired and self-repaired AI behaviors.

Existing work in the area as described in Chapter 2, section 2.4 focusses on using a library of pre-defined patterns of erroneous interactions among reasoning steps to recognize failures in the story understanding task [28]. This work requires construction of pre-defined patterns of erroneous interactions and a library of plans to repair the faulty behavior. As a result, the designer of the system h as to think of all the possible problems and solutions (plans) for them at design time. The repair approach presented in Chapter 5, uses the information available in successful and unsuccessful traces (created based on interactions with author created behaviors) to identify and address the possible failures. As a result, the possible ways in which the failures could be addressed depends upon the richness of other authored behavior sets (and generated execution traces for them), instead of being pre-crafted at design time.

Existing work in model based diagnosis [109, 105] uses a model of the system to localize the error in the system's element and uses the model to construct a possible alternative trace which could lead to the desired but unaccomplished solution. The approach requires an abstract model of the system where one defines a description of a real world system. The limitation of such an approach for repairing the behavior sets is that it requires a detailed model of the system in order to identify the problems with the constructed behaviors. The repair approach presented in this thesis doesn't require a detailed model of the system but instead provide a unique way of detecting the failures through use of knowledge available in successful and unsuccessful traces. Chapter 6 provides evidence of usefulness of AI repair approach in improving the constructed behavior sets.

## *1.9 Summary of Dissertation*

Chapter 1, implicitly motivates the research presented in this thesis. It presents the problem, formulated research questions and the solutions devised to address these research questions. The chapter also presents thesis contributions and claims. Chapter 2 provides background work related to several key areas of Game AI and in particular to AI behavior construction and repair. It presents (i) an overview of the state of art in game AI research, (ii) overview of research related to behavior construction and repair. Chapter 3 presents the first design pass over the representational vocabulary and the authoring activity that the novice user needs to conduct based on literature knowledge and personal experience. The chapter also present results from a user study conducted to look at whether the user would be able to understand the representational vocabulary and carry out the authoring process and what are the points at which simplification is required. Design insights from Chapter 3 are incorporated in the current computational authoring system. Chapter 4 presents the authoring system and the interaction design approach, representation vocabulary and execution

system. Chapter 5 presents the AI repair approach that is used to repair the behavior sets. Chapter 6 presents results from the four step evaluation study conducted to measure the success of the authoring system in allowing novices users to create the behavior sets. Lastly Chapter 7 summarizes the contributions of the dissertation as well as includes a discussion of the future work.

# CHAPTER II

# BACKGROUND AND RELATED WORK

In this chapter, I discuss some background work related mainly to AI behavior construction and repair. I first present the guidelines for novice user environments based on work in other fields like novice programming and end user product development to understand the complexity of the problem of creating an authoring environment for novices. I then discuss some of the work done in creating authoring environments in various fields and then look at some of the traditional work in AI behavior repair and situate my work within it.

## *2.1 Guidelines for creating an authoring environment for novice users*

Lets look at some the guidelines for an authoring environment that have been proposed in other fields to understand the complexity of the problem of creating an authoring environment for novices.

I. **Rapid prototyping**: One of the essential requirements to support a beginning user is to allow him to quickly see his creation. Novices should be able to carry out iterative design using rapid prototypes [93]. Rapid prototyping provides the novice users the ability to quickly create behaviors and constantly critique, adjust, modify, and revise them.

II. **Exploration and visualization**: Computational systems that support novice users should facilitate easy exploration of alternatives and enable easy backtracking. The novice user should further be provided the facility to compare results in what-if scenarios to enable him to easily separate what will stay fixed

from what should vary. Further, he should be able to get immediate feedback on his designs through visualization of behaviors in a simulation environment. Immediate feedback aids problem solving and has been an important factor in the success of spreadsheets [66].

III. **Understandable Metaphors**: When designed for novices, the system should speak the users language, with words, phrases, and concepts familiar to the user. When the metaphor is good, users can infer how the programming system works by referring to their existing knowledge and expectations about how the modeled system works. If metaphor is conceptually close to, a concrete real-world system, it would make it easier for novices to carry out the task.

IV. **Low viscosity**: Viscosity is a measure of how much effort is required to make a small change to the program [8]. As revision is an inherent part of programming, the environment should support easy and fast modification of novice constructions [31].

V. **Minimize working memory load**: Novice users rely on working memory while programming. As a result, if the environment doesn't support minimizing their working memory load, their performance suffers [6].

VI. **Visual vs Textual**: In designing the authoring environment, there is a debate on whether text or visual environment is better for supporting novice users. There is a widespread tendency to expect visual languages to be superior to text for novice programming, referred to as graphical superlativism [45]. There are other studies, however, that have reported that graphical superlativism does not hold in larger more complex programs [42] and at times it take longer to understand things in graphical than textual ones [46].

VII. **Common library**: Novice users can be helped by providing a library of standard modules that they can build upon. Provision of such standard modules reduces the starting effort for the users. These standard modules allow them to focus their effort on those aspects that they really want to customize or adapt [84] .

VIII. **Debugging support**: Novices who are able to program do not automatically gain debugging skills in the process of learning to program, suggesting that it would be useful for debugging strategies to be taught or for the environment to be proactive in suggesting strategies [84] .

IX. **Documentation support**: Even though it is better if the system can be used without documentation, it may be necessary to provide documentation. Examples and analogies play an important role in understanding [66].

X. **Social support**: A social context where a community is involved in carrying out the design activity provides necessary feedback to novices. Novice designers can participate in online communities where they can communicate, exchange ideas and software that extend the design from its original shape. Carrying out the activity with the group make it much more fun. Since people often learn more effectively in groups, perhaps it may help the initial learning process to provide a social context in which design can occur [84] .

A conclusion that can be drawn from the above list is there are multiple dimensions along which support could and perhaps needs to be provided to the novice users. Novice users are typically neither motivated nor skilled to modify and adapt the systems at the same level as software professionals. Therefore, it becomes important to provide tools that empower users to continuously adapt their systems at a level of complexity that is appropriate to their individual skills and situation. Advent of newer technologies like open source, free software and other tools that are accessible and

affordable to the general public, has opened up the way for novice users to generate content that is publicly available and is produced by novice users. With the advent of Web 2.0, this activity has increased considerably. Next, I look at the phenomenon of content generated by users facilitated by these newer technologies.

## *2.2 Advent of user generated content*

Media content development has traditionally been restricted to the experts who are proficient in the concerned domain. Technologies like open source, free software and other tools have has allowed novice users to participate and generate content that is publicly available. The concept has been given many names: prosumer [106], produser [16], user generated content (UGC), peer to peer, the former audience [41] and participatory culture [50]. The growth of UGC has been seen for many media forms, namely: photos and videos, wikis, blogs, discussion board, product reviews among others [17]. UGC has been in a latent form in one way or another since the very early days of Internet. However, in the past few years, it has become one of the prominent forms of global media. One of the primary reasons for this phenomenon is opening up the tools of production (also called democratizing the tools of production [5]) in any domain to everyday users. In the area of virtual worlds, games and interactive stories as well, UGC has been one of the prominent ways of expanding the content [64, 33, 5, 76]. Commercial games producers understand the importance of user contribution and integrate user creations into their production processes with increasing frequency [79] and is considered the first industries to place emphasis on the idea of systematically empowering its customers [85, 52, 51]. Users are involved in the process through editing game levels etc, as a result making the game products become more useful. AI authoring in these games is carried out through scripting languages [107, 77]. Creating AI behaviors using these languages require specialized skills in programming and development. Although user-generated game content has

been around for years, AI programming in these game mods have tended to come from people with computing skills well beyond the average user. The challenge for those seeking to make game authoring more accessible has been to find a way to make AI authoring powerful without being complex, which turns out to be hard to do than it might seem.

## *2.3 Related work AI behavior construction*

This thesis is motivated towards a better scientific understanding of the simplification needed to enable the development of future environments for AI behavior authoring that can effectively help novice users carry out AI behavior authoring process. There have been other works in academic circles with similar goals. Let us next discuss some of the work done in the area of development of authoring tools in general for various different domains with the focus on interactive storytelling.

### 2.3.1 Authoring Tools

#### *2.3.1.1 Storyboarding authoring tools*

The development of authoring tools has also been explored by the multimedia community. The research in this area has mainly focused on supporting authors and designers in authoring digital media tools and especially multimedia presentations and multimedia documents. Typically, these tools provide controls for user interactions, sequencing utilities and a scripting language. Adobe Director[1] and Authorware Creations[2] allow users to create and publish interactive games, demos, simulations, and eLearning courses for the web. With both of these tools, video and graphical elements can be arranged with score and/or a timeline. A graphical interface makes it possible to edit multimedia presentations and Adobe Director's scripting language allows fine control of user interactions. Research into developing rapid prototyping

[1]http://www.adobe.com/products/director/
[2]http://www.adobe.com/products/authorware/

tools that facilitate the preproduction process has led to the creation of DEMAIS [10]. This tool addresses the need of multimedia designers for early behavior exploration in an attempt to expand their ability to quickly investigate and effectively communicate behavioral design ideas. DEMAIS is a sketch-based, interactive multimedia storyboard tool. It uses ink strokes and textual annotations as an input vocabulary that is employed to transform static sketches into working examples. Jhala et. al. also describe an intelligent storyboarding tool that allows automatic construction of dynamic storyboards based on an XML based input scheme specifying action sequences and abstract camera directives[53]. These storyboarding tools cannot be used for creating AI behaviors however, the work done in this area is very useful and was worth paying attention in my pursuit of creation of tools for AI behavior authoring for novice users as storyboarding tools are very accessible for novice designers without any programming experience.

#### *2.3.1.2 Interactive storytelling authoring tools*

Authoring tools have gained popularity also in the research area of storytelling and interactive fiction. The vision driving the creation of authoring tools for interactive narrative is commendable since it explores the difficult task of making users play digital games with great and challenging plots as well as life-like, believable characters. In this specific domain, authoring tools are also very helpful since they can be used to create the structure of non-linear tales which allow to interactively play out different variations of a story.

Agent Stories [15] is an early approach to the creation of multi-threaded stories and audio/visual narratives. In Agent Stories, the designer defines the structure of a story by providing links between clips. Links are therefore used to define relationships between different elements and plots of a story and results in a sort of story web which can be traversed differently to generate different narrative paths.

The Viper project [22] is a tool that allows the creation of responsive video programs from a database of video clips. The user can express a structure by defining relationships between clips and by specifying properties and constraints. Viper also lets users create complex video sequences that can re-organize themselves in response to a set of audience-related factors.

The Interactive Storyboard [113] is a prototyping application targeted at novices, and especially children, that helps them create visual stories in a quick and easy way on a touch interfaces. The initial conceptual design of an authoring tool tailored for a specific narrative agent architecture that powers a virtual bullying drama was presented in [58]. The same group of researchers, also proposed the experimental platform ENIGMA for collaborative authoring of virtual character behaviors [59]. ENIGMA tackles the problem of knowledge acquisition for generative storytelling using a combination of crowd-sourcing and machine learning techniques. Other similar authoring systems like INSCAPE [114] and U-Create[95] also let authors choose and modify the configuration of the environment peculiar to the narrative such as the characters' positions, the place of objects, etc. These systems however do not make it possible to create the narrative content. The creation of this latter is one of the focuses of SWAT, the authoring tool for the storytelling engine Storytron [29]. SWAT is pre-loaded with a small number of storyworlds for the novice users to tweak. An empty storyworld to build from the ground up is also provided. Unfortunately, SWAT suffers from a rather complex interface that makes the process of creation of plots very similar to coding using a visual programming environment.

Wide Ruled [100] is an AI-based authoring approach to interactive textual stories. It is aimed at novices with many different backgrounds and implements a generative plan-based approach which is based on the UNIVERSE [65] author-based model of planning. In Wide Ruled a story is a representation of a plan that is carried out within the context of a specific story world. The story space is modeled as a set

of goals specified by the authors. Each goal itself can have one or more plans that represent a plot fragment which can fulfill the goal. Plans are activated according to a set of preconditions. Story Canvas [99] is an evolution from Wide Ruled since it expands its scalability for the authors to be able to manage larger story worlds. StoryCanvas provides a visual authoring interface to an underlying plan-based representation of interactive stories and exploits graphical metaphors for presenting the complex constraints and actions in the underlying story model.

A relatively recent overview of related work and authoring tools available can also be found in [72]. ScriptEase is another tool that allows the users to create story based game through a pattern catalog, identified through rigorous analysis. In order to increase usefulness of the system to new stories and domains which cannot be covered by existing patterns, the authors need to design new patterns [71].

Good and Robertson describe another approach towards creating interactive stories. They use Neverwinter Nights game toolkit for allowing the creation a narrative-based game[44]. Teatrix is another approach towards story creation where a 3D virtual environment allows children to cooperate in the creation of their stories, each one taking the role a character [88].

Work in the area of authoring tools for interactive storytelling is focused on providing authors the necessary tools to create stories. My focus in this dissertation work is in particular towards creating AI behaviors for the characters who are part of a story. Repairing the constructed AI behaviors by providing AI scaffolding in the authoring interface is another important aspect of our work that distinguishes my work from the story authoring work. In the case of novice users, supporting them in debugging these constructed artifacts (AI behaviors in our case) becomes very important as they do not have the necessary background to identify problems with the created behaviors.

#### *2.3.1.3 Authoring tools for teaching programming*

Scratch [48, 17, 64] is a programming language that allows for quick and easy creation of interactive stories, games, and animations to share on the web. The grammar of Scratch is based on a set of graphical programming blocks that can be put together to create complete programs. The Scratch user interface also has a scripting area which can be used as a physical desktop where users can leave blocks for future use thus allowing for an experimental approach to authoring that parallels an iterative and incremental design.

Alice [56] is another programming environment designed to motivate and enable middle school female students to learn to code through the creation of short 3D animations in a fun and engaging way. Alice comes with a set of high-level animations that makes it possible for users to program social interactions between different characters. A gallery of 3D characters and scenery with custom animations is also provided in order to stimulate and incite ideas for new stories. Students find Scratch to be very accessible, and can do many cool things very quickly. The downside is that they will hit the limits of Scratch relatively soon. Due to their similarity, there has been several discussions concerning the relative merits of Scratch and Alice. In general, students find that Scratch is very accessible and allows to create digital content very quickly. On the other hand, Scratch is relatively limited in its capabilities. Alice instead offers a wider set of features and capabilities but students need a longer training to utilize them in full.

Scratch and Alice set up the authoring task as a programming activity and are not specifically designed for authoring AI behaviors for characters. The cognitive gap between the programming activity and AI authoring activity makes it harder to use them for authoring AI behaviors. The cognitive gap becomes more difficult to bridge when the targeted user base is novice users. Moreover these tools do not provide any debugging scaffolding to the authors which for novice user is important as they do

not have the necessary background to identify problems with the created artifacts (AI behaviors in our work).

#### *2.3.1.4 AI behavior authoring using learning from demonstration*

Authoring of AI behaviors by assisting the authors is also reflected in the field of "Programming by Demonstration" [30]. The central theme in this work is to allow the users to instruct the computer on how to perform a certain task, instead of writing a program to solve it. The system observes the users actions and learns from it to perform those actions by itself. Nicolescu [78] describes a modular architecture which allows a robot to learn by generalizing information received from multiple types of demonstrations, and allows the robot to practice under the demonstrator's supervision. Floyd et. al. present an approach to train a Robocup soccer-playing agent by observing the behavior of existing players and determining the spatial configuration of the objects the existing players pay attention to [36].

The approach has been used to author game agents in computer games as well. An example is the restaurant game developed by Jeff Orkin [82] at the MIT media lab. The game consists of two players a waiter and a guest in the restaurant. The player can take any of the roles. The game is freely downloadable for the public and all game sessions are logged. As users play the game taking the role of a waiter or a guest, more logged sessions are recorded. The system learns from theses logged game traces and the resulting learned behaviors are used to power AI characters (taking up the role of a waiter or a guest). In another system called D2 [81], author demonstrations are used to learn behaviors for a Case based planning agent playing a real time strategy game Wargus.

Learning from demonstration(LFD) [14] is a promising approach for authoring AI behaviors. The approach can be an important part of an integrated authoring framework where the authors can use it to author initial version of the behaviors

and can then go in an authoring interface to modify the behaviors in more detail. Approaches mentioned above, however just use LFD as the sole way of authoring behaviors for robotic or game agents. Using learning from demonstration as the sole approach for authoring behaviors takes away the authorial expressivity that comes when using an authoring environment where the authors could have more fine-grained control over the authoring process. Moreover these approaches do not tackle the issue of repairing the problems in the constructed behavior sets (through learning from demonstration) and supporting authors in debugging the problems in the authored behavior sets.

#### *2.3.1.5 AI behavior authoring in computer games*

Simple finite state automaton are typically used to author game characters [91] by the game companies. While finite state automata are easy to develop, they are predictable. Over long game sessions, a character's static behavioral repertoire may result in repetitive behavior which harms the believability of the characters [94]. Therefore, many developers of computer games and robotic toys have turned to hierarchical, probabilistic and fuzzy state machines [96]. The advantage of these systems is that layered control enables authoring sophisticated behaviors, and probabilistic transitions makes the actual behavior of the agent nondeterministic, less predictable, and more realistic.

Other game developers have turned to scripting languages which allow arbitrarily sophisticated behaviors. Common scripting languages, such as Python [63], have been adopted by some companies for AI scripting [9, 37, 38]. Some other companies have used Lua [62] for scripting AI behaviors [24, 108, 34]. These scripting languages allow arbitrarily sophisticated behaviors [75]. However, creating AI using scripting languages can be very labor intensive and is still fundamentally a programming activity that makes it harder for the novice users who do not have the programming

expertise to carry it out. Computer game manufacturers typically employ dozens of engineers to perfect very simple game AI(for examples, see the Postmortems in Game Developer Magazine, e.g. [49, 101, 80]). The static nature of the scripted behaviors means that when the behaviors fail to achieve their desired purpose, the game AI is unable to identify such failure and will continue executing them. My work supports novice users by providing a repair layer for the authored behavior sets that identifies failures in them and repairs them to improve their performance.

In the area of virtual characters, Sims 3, a commercial game, allows their end-users to create new characters using predefined personality traits [4]. Creating personality through such a mechanism on one hand makes the job of the end-user to create characters easier, however, on the other it limits user's creative freedom. My focus is on allowing the end-user to have more creative freedom by allowing him to create new personality elements (in the form of AI behaviors). This is a difficult task, as it requires fundamental understanding of the simplification needed to make the process novice user understandable, as well as an understanding of the necessary scaffolding required in a tool aimed at the creation of AI behaviors by novice users.

Another approach is reflected in using behavior trees to represent behaviors for interactive characters[90]. Behavior trees are a combination of linear scripts and reactive state machines, and provide the goal-directedness of planners. REDUX provides another approach where the Subject Matter Experts (SME) specify behaviors using abstract scenarios represented as diagrams. SME's graphically describe the conditions under which actions and goals should be pursued along with the associated reasons for those decisions. The system then analyzes and generalizes from the example scenarios, and alerts the SME to inconsistencies and missing knowledge. The approach used is promising, although without a detailed evaluation with a variety of novice users, its hard to judge whether the approach could be used easily by them. The identification of problems in the authored behavior sets is also limited. The inconsistency check

in REDUX is limited to problems in the preconditions of the behavior checking for example, whether these preconditions are overly general or specific. It does not check for other kinds of problems in the authored behaviors like problems with the actions sets that are defined for a given behavior or does not provide any kind of suggestions on newer kind of behaviors that might be missing from the existing behavior sets which as detailed in Chapter 5 is supported by the repair approach presented in this thesis.

Next, I present some of the related work in AI behavior repair. First, I present theoretical foundation of behavior repair and discuss the limitations of some of the prior work with respect to the problem of behavior repair in interactive games. I next present existing work in the area followed by related work in introspective reasoning.

## *2.4 Related work on behavior repair*

### 2.4.1 Theoretical foundation of behavior repair

The agent's behavior set can be considered a reactive plan dictating what it should do under various conditions. Behavior repair can then be considered a problem of reactive-plan revision. One approach to plan revision is to simply apply classical planning techniques to replan upon encountering failure. Such techniques, however, are ill-suited to the unique requirements of a game domain. They typically assume the agent is the sole source of change, actions are deterministic and their effects well defined, that actions are sequential and take unit time, and that the world is fully observable. In an interactive, real-time game domain, all these assumptions are violated. Actions are non-deterministic and their effects are often difficult to quantify. Game domains are typically not fully observable. There are often occlusions blocking sensors from reaching the entire world.

Some of the more recent work in planning has focused on relaxing these assumptions. Conditional planners such as Conditional Non Linear Planner [86] and Sensory

Graph Plan [112] support sensing actions so that during execution, changing environmental influences can be ascertained and the appropriate conditional branch of the plan taken based on the sensor values. Unfortunately, as the number of sensing actions and conditional branches increase, the size of the plan will grow exponentially. These techniques are mostly suited to deterministic domains with occasional exogenous or non-deterministic effects, not to continuously changing interactive domains.

One of the approaches to planning that deal best with exogenous events and non-determinism are those based on Markov decision processes (MDP), focusing on learning a policy. These approaches, however, require a large number of iterations to converge and only do so if certain conditions are met. In complex game domains, these techniques are intractable (MDP learning algorithms require a polynomial time in the number of states of a problem [55], which in the case of complex games is prohibitively large). Physical states alone are complex, upon adding game state information, the status level and internal states of various characters, the state space quickly grows untenable. Further, these approaches generalize poorly. An interactive player can significantly change the game world; the learned static policy will have to be retrained to accommodate such changes.

In the AI planning community, there has been previous work on techniques for combining deliberative and reactive planning. For example, Atlantis [39] and 3T [13] are all aimed at combining deliberative and reactive components. Unfortunately they all, to varying degrees, make classical planning assumptions and are thus not applicable to a real-time interactive games. These approaches, furthermore, treat reactive plans as black boxes; planning sequences of black-boxed reactive plans, but not modifying the internals of the reactive plans themselves.

Behavior-set rewriting can be cast as a transformational planning problem. In transformational planning, the goal is to improve an existing reactive plan by applying a set of plan transformations. [70] describes such an approach where the agent tries

to improve the expected utility of its plan in a world where its job is to transport balls from one location to another through an obstacle-filled space. More recently, other transformational planning approaches have used temporal projection of a robot's plan to detect problems with the plans using a causal model of the world to represent the effects of their actions [12]. These approaches, although promising, are of limited usefulness in game domains. They require a detailed casual model of the world. In games domains, there is neither the time for extended projective reasoning nor can accurate projection be performed due to the interactive and stochastic nature of game domains.

### 2.4.2 Metareasoning

*Metareasoning* is the process of monitoring and control of reasoning. The control part of *Metareasoning* involves the meta-level control of computational activities dealing with question of when to stop a given computational process. The monitoring part involves figuring out what went wrong and the reasoning behind it through the process of introspection also called *introspective reasoning*. The system presented in this thesis explores the later aspect of *Metareasoning*. There have been various approaches to *metareasoning* and more specifically to the task of *introspective reasoning* presented by various researchers [27, 7]. Meta-Aqua system, for example, by Cox and Ram [28] uses a library of pre-defined patterns of erroneous interactions among reasoning steps to recognize failures in the story understanding task. This work requires construction of pre-defined patterns of erroneous interactions and a library of plans to repair the faulty behavior. As a result, the designer of the system has to think of all the possible problems and solutions (plans) for them at design time. Using the approach for repairing the behavior sets in our work would have required us as the designers of the system to think of all the possible way in which the behaviors could be repaired once the failures are identified. Instead, in my approach the system uses the information

contained in the execution traces (created based on interactions with author created behaviors) where the failures do not exist to address the possible failures. As a result, the possible ways in which the failures could be addressed depends upon the richness of other authored behavior sets (and generated execution traces for them), instead of being precrafted at design time.

Autognostic system by Stroulia and Goel [105] uses a model of the system to localize the error in the system's element and uses the model to construct a possible alternative trace which could lead to the desired but unaccomplished solution. The approach requires an abstract model of the system where one defines a description of a real world system. The model explicitly represents the structure of the system i.e. its constituent components and their connections. The problem is detected when an observation of the system 's actual behavior conflicts with the system's expected behavior. The approach uses the model of the system to backtrack and find the fault point. The limitation of such an approach for repairing the behavior sets is that it requires a detailed model of the system in order to identify the problems with the constructed behaviors.

The above approaches have generally been applied to detect failures in plans consisting of STRIPS operators or minor extensions of STRIPS; using it for revising complex behavior sets for game domains requires approaches that can deal with the complexity of the behaviors. Furthermore, these approaches are not designed for real time domains, they assume that actions are deterministic and their effects well defined, that actions are sequential and take unit time, and that the world is fully observable. In an interactive, real-time game domains, all these assumptions are violated. Characters are constantly interacting with the user, actions are non-deterministic and their effects are often difficult to quantify.

More recent work has relaxed these assumptions and has been applied to more complex game domains. Introspect system, for example, observe its own behavior so

as to detect its own inefficiencies and repair them. The system has been implemented for the game of Go, a deterministic, perfect information, zero-sum game of strategy between two players [20]. Ulam et. al. present a model based *Metareasoning* system for FreeCiv game [109] that uses a self-model to identify the appropriate reinforcement learning space for a specific task.

### 2.4.3 Adaptive AI

Another body of related work is in Adaptive AI [102]. Dynamic Scripting is a technique based on reinforcement learning, that is able to generate 'scripts' by drawing subsets of rules from a pre-authored large collection of rules [103]. If the rules are properly authored, different subsets of those rules provide meaningful and different behaviors. The technique is based on learning which subsets of those rules work better under different circumstances, by learning which rules work good or bad, increasing or decreasing their probability of being selected again. The approach assumes that the initial authored rules are perfect and the goal of the adaptive AI is to learn combination of the rules that would work good or bad for a given circumstance. In my work, where the goal is to adapt/repair the authored behavior sets, the behaviors can be thought of as equivalent to rules in the adaptive AI work above. In finding issues with the authored behavior sets, the behaviors cannot be considered as black-boxes (as is the case with rules in the adaptive AI work above) and assumed to be perfect. The adaptive AI work only learns conditions for the applicability of the different rules. It is unable to identify any problems with the internals of the rules.

Adaptive Behavior Language (A2BL) provides another approach towards adaptive AI where the adaptive programming primitives are directly built into the ABL language [98]. The approach supports partial programming, a paradigm in which a programmer needs to only specify the details of behavior known at code-writing time, leaving the run-time system to learn the rest. The approach is based on having

reinforcement learning integrated in the ABL language itself. The approach though promising, has the following limitations with respect to using it for a novice user authoring environment. First, it requires novice authors to learn a programming language to allow the characters to be adaptive. Second, adaptation of characters authored in the language to changing circumstances requires a lot of examples, as is the case with any reinforcement learning based approach.

## *2.5 Chapter Summary*

In this chapter, I have discussed background work related to AI behavior construction and repair. I have presented the guidelines for novice user environments based on work in other fields like novice programming and end user product development to understand the complexity of the problem of creating an authoring environment for novices. I have also looked at some of the work done in creating authoring environments in various fields like storyboarding, interactive stories and computer games and looked at some of the traditional work in AI behavior repair and situated my work within it.

# CHAPTER III

# AI BEHAVIOR AUTHORING: PAPER PROTOTYPE BASED FIRST USER STUDY

This chapter explores the first research question presented in Chapter 1 i.e. the design solution that would enable novice users to successfully construct AI behaviors. In this chapter, I detail on the first design pass over the representational vocabulary and the authoring activity created to identify the right user interaction approach. I also discuss a paper prototype version of the authoring interface that the novice user used to create behaviors. This paper prototype version was created based on personal background in creating interactive experiences with AI characters[1], knowledge of using some of the reactive architectures[2] and a thorough literature survey. In this chapter, I also present results from a user study conducted to evaluate the paper prototype version of the authoring activity. The goal of the study was to understand whether the users are able to conduct the authoring process using the paper prototype version and understand the representational vocabulary. The goal of this study was to identify and modify any early design flaws.

One of the key results that came out of the study was that some of the existing interaction design approaches, representation formalism and vocabulary that we employed in the authoring interface were unsuitable for use when creating an authoring environment for novice users. As AI experts, the first instinct was to conceptualize

[1]Before coming to Georgia Tech, I was involved in a project creating conversational interaction with fairytale author Hans Christian Andersen. The project involved demonstrating universal natural interactive access, in particular for children and adolescents, by developing natural, fun and experientially rich communication between humans and embodied historical and literary characters from the fairy tale universe of Hans Christian Andersen [25, 26]

[2]ABL, one of the reactive language for believable characters was used in one of our previous work done at Georgia Tech [115]

the behavior authoring process as a hierarchical top down activity for novice users. The results from the study indicated that novice users had difficulty conceptualizing the behavior creation process as a hierarchical, top-down process. Here is a short, not exhaustive list of the relevant findings from the user study:

- It is difficult for authors to conceptualize the behavior generation task in a hierarchical top-down structure. Authors were more attuned to thinking of the behavior creation task in a sequential manner.

- In order to concretely see the effects of their created behaviors, there was the need for providing concrete feedback to the authors. This would help them identify problems with the behaviors directly during the authoring task.

- As part of the behavior creation process, authors were required to fill in various behavior conditions (e.g. conditions for activating the behavior). The results indicated that the users had difficulty creating some of the behavior conditions necessitating the need of simplifying some of these conditions.

- As part of the behavior creation process, users also needed to define the behavior conditions in terms of a basic set of events that could be detected from the concrete world. The study revealed that the steps required to link these behavior conditions to set of events that could be detected from the concrete world needed to be simplified. The users seemed to have a lot of difficulty in linking behavior conditions to lowest level events detectable from the world.

## *3.1 Representational Vocabulary and Authoring Activity: First Pass*

### 3.1.1 Representation Vocabulary

In order to identify the basic vocabulary and terminologies, existing reactive architectures were used as a starting point. The key property of reactive architectures

is their reactivity and responsiveness to time-critical events, such as emergencies in a robot assistance domain, or dodging and avoiding unexpected obstacles in real-world navigation domains. This reactivity to time critical events makes them ideally suited for creating interactive avatars. Examples of reactive architectures include Firby's Reactive Action Packages [35], Kaelbling and Rosenschein's Gapps formalism [54], Georgeff and Lansky's Procedural Reasoning System [40], Newell and colleagues' Soar architecture [60]. Another example of reactive architecture used specifically for interactive characters is the A Behavior Language (ABL) created by Michael Mateas. ABL is a proven language for believable characters, which has been successfully used to author the central characters Trip and Grace for the interactive drama Facade [68]. By analyzing these existing reactive architectures and based on personal experience creating interactive characters, I identified the following terms. Notice that the definitions of these terms described here do not intend to define them in general, but are described here to clarify the way in which they have been used for the purposes of the authoring interface.

- **Scene:** provides a social context under which avatar behaviors can be described. An example of a scene might be "a boy meeting a girl at the park for a date". Scenes may contain one or more characters, and can be of arbitrary length.

- **Behavior:** represents the activities that the avatar would/can accomplish during his interactions. Behaviors can be considered similar to plans in a traditional AI planning paradigm. For example, as part of the scene "a boy meeting a girl at the park for a date" described earlier, the boy avatar could perform a "flirting with a girl" behavior. Behaviors can consist of a sequence of other *behaviors* and *actions*. Behaviors can exist both at a higher levels and lower levels. For example, "flirting with a girl" is a higher level behavior that could contain "approach girl" behavior at a lower level.

- **Action:** are performed concretely in the world by the avatar. These include gestures like laughing, talking, wave etc. These are considered as the lowest level behaviors that can be performed by the avatar in the world and cannot be further decomposed. They represent the elementary units on which all behaviors are further built.

- **Personality:** represents character's overall way of conducting himself. These are manifested in the way a specific avatar goes about performing the scenes through the kind of behaviors he tries to pursue, the way he carries out these behaviors and the way he conducts the basic actions.

- **Emotion:** represents the avatar feelings in response to things happening in the environment.

- **Percepts:** represent basic things that the avatar detects from the environment.

- **Likes/Dislikes:** represent things that the avatar likes or dislikes.

Behaviors are further associated with the following conditions.

- **Pre-Condition:** detects when the behavior needs to be activated and carried out.

- **Success-Condition:** detects when a behavior has been successful. This information is needed to identify when a behavior needs to be pursued again.

- **Non-Progress-Condition:** detects when a behavior is not progressing well. If a behavior is not progressing well, the character would try to repair the situation by different means. (like pursuing a different behavior to achieve the behavior or have an emotional reaction in response).

- **Failure-Condition:** detects if the behavior has failed to achieve its purpose.

- **Advancing-Condition:** keep a check on whether the behavior is progressing well or not.

Please note that *Success-Condition* and *Failure-Condition* can be thought of as opposite of each other. They were used separately to provide more flexibility to the user to use either of them as needed. At times, it is easier to think in terms of one than the other. Similar reasoning went into providing *Non-Progress-Condition* and *Advancing-Condition* as separate entities. As part of the authoring process, these conditions are created by the user using basic sensory information through mathematical and logical operations that can be detected from the virtual world.

I decided to employ Second Life as the concrete virtual world in which the behaviors can be executed. Second Life provides a comprehensive list of basic actions that the character can perform [2] and basic set of sensory information that can be detected from the environment [3]. In order for the character to carry out the created behaviors, the behaviors need to be defined down to the level of granularity of basic actions. Therefore, as part of the authoring activity, the novice user defines each behavior in terms of its basic actions. Likewise, in order to detect the conditions associated with the behaviors, these conditions needed to be defined in terms of the basic set of events that can be detected from the concrete world.

Next, we discuss the authoring process that we envisioned the authors to carry out. The authoring activity was derived based on authoring various interactive characters using reactive planning languages[3].

[3]As described earlier, before coming to Georgia Tech, I had been involved in creating conversational interaction with fairytale author Hans Christian Andersen. Experience of authoring interactive characters in Hans Christian Andersen domain [25, 26] and various other projects at Georgia Tech [115] helped in deriving a first pass at the user's authoring activity presented here

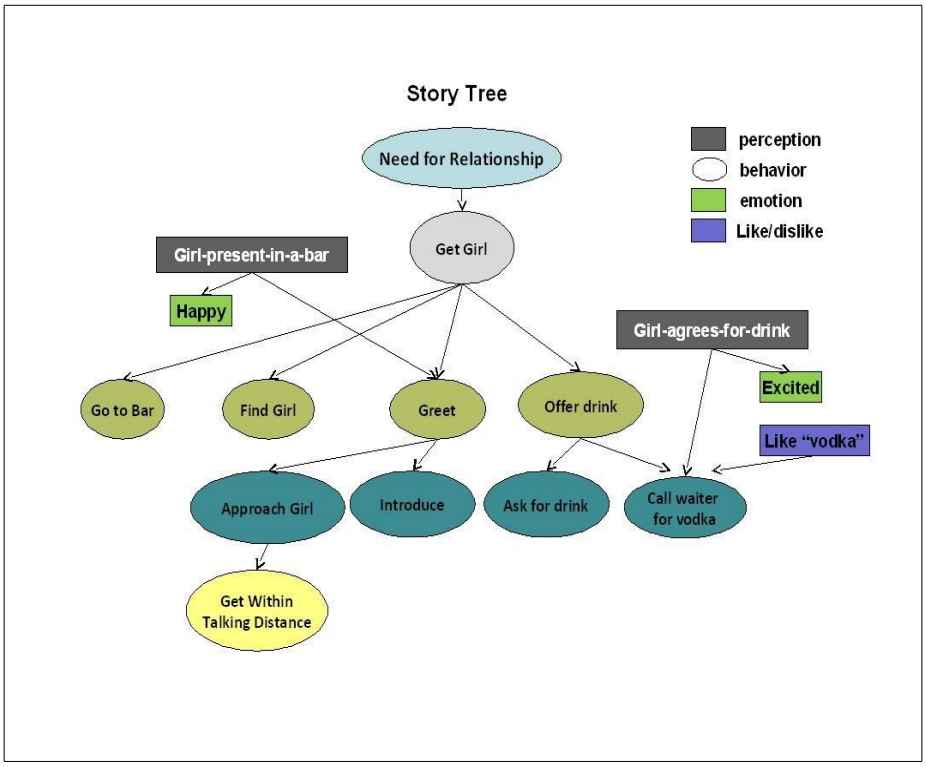



**Figure 2:** The figure shows the behavior tree structure created for a scene

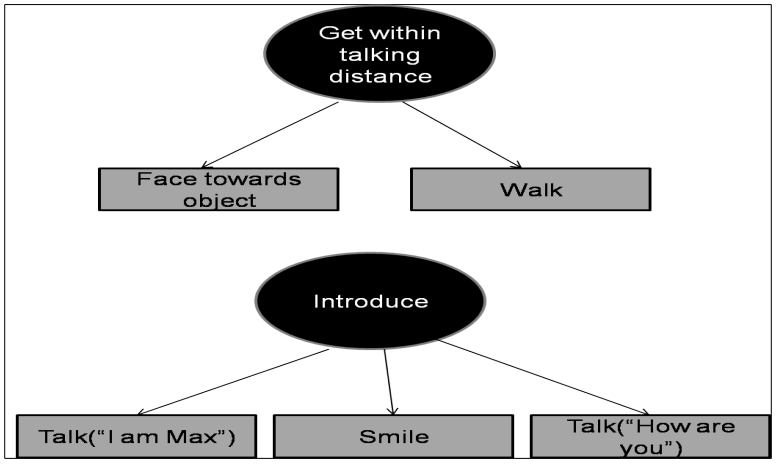



**Figure 3:** The figure shows the behaviors at the lowest level of the tree linked to the basic actions available from the virtual world

### 3.1.2 First pass: The Authoring process

The authoring process that we envisioned the author's to carry out consists of the following three steps:

(I.) The first step involves scene creation with hierarchical behavior structure. Authors create a hierarchical structure of the things a character would carry out. Scenes may contain one or more characters, and can be of arbitrary length. Consider a scene of a "boy flirting with a girl in a bar". The given scene consists of two avatars, a girl and a boy. The boy has a personality trait of being "flirtatious" and "aggressive". The female has a personality trait of being "shy" and "looking for relationship". For such a scene, the behavior tree is shown in Figure 2. The top-down structure linking lowest level behaviors in terms of basic actions that can be carried out in the virtual world is also defined by the authors as shown in Figure 3. For the behavior "Introduce", three actions are defined namely, a) Talk action with parameter "I am Max", b) Smile and c) Talk action with parameter "how are you". Figure 3 shows two example behaviors linked to the basic actions.

(II.) The second step consists of defining the various conditions for the behaviors as mentioned in subsection 3.1.1. Various conditions for two example behaviors are shown in the Figure 4. For example the behavior "Go to Bar" is associated with an activation condition that the boy wants to chill out (defined using the tag "want to chill").

(III.) The third step involves linking higher level conditions with the basic level percepts available from the world. For example, the condition "girl present in a bar" can be detected by performing an inside operation on location of bar and location of the girl ( as shown in Figure 5). The sheets containing list

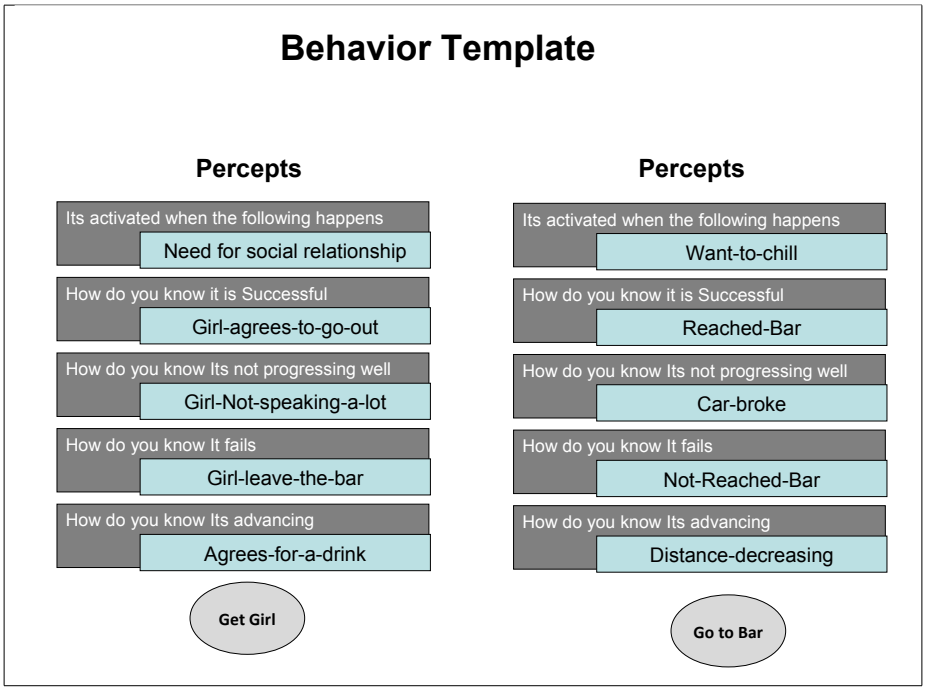



**Figure 4:** The figure shows conditions for two behaviors

of basic actions and percepts that were provided to the authors is shown in Figure 73 and 74 in section A.1.

## *3.2 User Study: Paper Prototype based Authoring Environment*

Next, some simple paper prototypes were created based on the authoring activity described above. Next, a user study was conducted using these paper prototypes to identify if the user would be able to understand the representational vocabulary and carry out the authoring process following the steps listed above. The study was guided by our broader research question of identifying the design solution that would enable novice users to easily construct AI behaviors. More specific research questions related to this broader question were the following:

- Which parts of the representation vocabulary are difficult for users to understand?

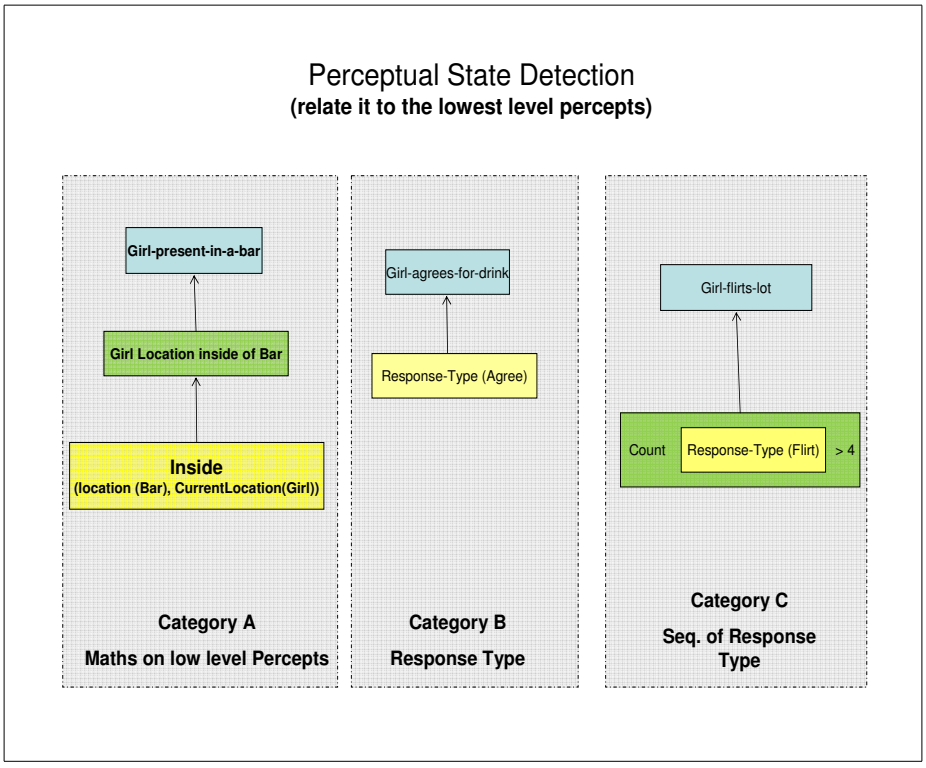



**Figure 5:** The figure shows highest level percepts to lowest level information available from the virtual world

- What parts of the authoring process are difficult for users to conceptualize?
- What are the points at which the authoring process becomes tedious?

### 3.2.1 Study Setup

Forty participants (referred as P1:P40) were recruited, 10 females and 30 males. The participants were pre-screened to make sure they did not have any programming and design background. The participants ranged across multiple races, education levels, and ages (from 22 to 40 with an average age of 28). Participants were paid $10 per hour rounded up to the nearest hour for their time. Players signed a consent form at the beginning of the session. The authoring tasks was described to participants through example forms (as shown in Figure 2-5). The participants were given a brief description of the scene setting. For example, they were told that the scene involves a boy who will take a girl to a prom. Two such scene settings were provided to the

participants. Scene settings were selected such that the participants would find them interesting. The first scene was a situation about a boy taking a girl to prom, while the second scene involved an armed robber who holds up a person. The list of basic actions and basic events that can be detected from the environment was also provided to the user (shown in Figure 73 and 74 in section A.1. The participants were asked to conduct the authoring activity using the paper-based prototype interface forms. These forms were blanked out versions of the filled forms shown in Figure 2, 3, 4 and 5. The users had to carry out Steps I-III outlined in Section 3.1.2. During the authoring session, a researcher observed the players interaction with the paper prototype and took notes. This included any unusual things such as "unable to fill a condition", or "having difficulty creating behaviors". On an average, a complete player interaction lasted for about 1 hour. An open ended interview about their authoring experience was conducted at the end of the session. The questions for the interview guide are listed in section A.2. Users were also asked to fill out a quantitative questionnaire at the end of their interaction. Quantitative questions are shown in section A.3.

### 3.2.2 Data Analysis

To perform the analysis, user responses from the interviews and observed user actions during the authoring process were transcribed. I analyzed the data obtained, focusing on a qualitative analysis of the results. In order to perform a qualitative analysis well-known qualitative analysis method, called Grounded Theory [104] was used. Using grounded theory principles, notes were made for user responses from the interview. The transcriptions were processed, first to take open-ended notes and second to highlight more salient ideas expressed by user. I generated codes, such as "User is unable to fill precondition" and "User was not able to complete scene creation process", and listed multiple user quotes for each idea in an Excel spreadsheet. Within the spreadsheet, I then conducted data analysis where related codes

**Table 1:** The table shows some of the codes used for data analysis in labeling the interview data

| **Category** | Player Comments |
|---|---|
| **Complain about lack of feedback** | It was hard to know how well things worked out in actual practice |
| **Hierarchical behavior creation problems** | It was hard for me to think of things in a top down fashion. |
| **Problems defining Behavior Conditions** | It was unclear how I needed to create formulas to group together these lowest level things. |
| **Feels system was technical** | It was definitely very technical. You need to create mathematical formulas and things, that made it technical and not creative |
| **Uses his own imagination** | I was imagining how things would go when someone does these things. |
| **Unable to fill some behavior conditions** | It was hard for me what to fill in for some of the conditions. |
| **Unable to identify problems on his own** | Sometimes its hard to figure out why things are going wrong. |

were grouped together. For example, all codes that had something to do with the problem of creating behaviors were initially grouped together into related concepts. I went through an iterative process of describing each concept and conceptually linking them to each other. I formed a single hierarchy of the concepts, and organized the same concept with a paragraph to describe each concept along with the top two to three quotes for it. This hierarchical organization of concepts served as an outline for the qualitative findings. A subset of the codes are shown in Table 1. I also analyzed the results from the forms filled out by the users during the authoring process to identify any further issues during the authoring process and to get a better understanding of issues pointed by them during the interview session.

### 3.2.3 Design Implications

The analysis of data collected during the study provided us with some key insights useful for the next version of the design. Let's discuss them.

#### *3.2.3.1 Difficulty conceptualizing hierarchical behavior trees*

As I discussed earlier, the first stage of the authoring process involved creation of hierarchical behavior structure. Users created a hierarchical structure of what things avatar would carry out. The data analysis indicated that users had difficulty conceptualizing the behavior creation process as a hierarchical, top-down process.

> *When you think of a scenario where X needs to happen, you can think of a few things that would happen, like A would happen and then B could happen and so on. Here (using the authoring interface) I am thinking of things in a different way. I am thinking like A would happen and then in order to accomplish A, I need to do these two other things and then I would have to break them down, that is kind of hard. (P16)*

When users were visualizing the creation process, it was very hard for them to think "in a top down fashion" [P19] and they were more used to thinking of things in a sequential fashion.

> *When we do things, I do them in a very sequential fashion, it would be hard for me to plan out like I would do this and then it would be doing these other two things which would then involve other activities (P23)*

These difficulty of creating hierarchical structures is confirmed by high ratings on the quantitative questionnaires. To the question of difficulty level of creating behaviors, users provided an average rating of 4.03 (0-difficult and 5-very difficult) as shown in Figure 6.

Novice users would bring a combination of real-world experiences and experience with other software that they usually employ to understand the behavior construction task. A user interaction approach that can a) present the behavior creation task in a way that is more closer to the way (i.e. in a sequential fashion) novice users would think of the behavior creation process and b) avoid asking them to create

deep hierarchical structures would help them in easily conceptualizing the authoring process.

#### *3.2.3.2 Provide for feedback on created scene*

Users commented that they had difficulty seeing the implications of their authoring process. Creating the behaviors on a paper prototype did not provide the users an opportunity to evaluate the created behaviors. Users mentioned that it would have been easier for them to create behaviors if the system could play "a simulation of the behavior"[P13] such that they could "verify or correct their creations"[P15]. P2 mentioned:

> *I was imagining how things would go when someone does these things, I could see that there could be some things which might not work exactly, I would say it would have been easier if things played out in a game, that would give good ideas on what works and what doesn't. (P2)*

Users mentioned that they would want to see how "the scene they had created would work in actual practice" [P10]. They felt that the scene "might come out funny or some gesture might not work" [P24]. Looking at the avatar acting out the created behaviors would be very helpful in finding how well things would work in actual practice. P19 mentioned:

> *Seeing my creation being acted out would be great. That would tell me how well it works from the performance perspective (P19)*

During the authoring process, it would be very hard for users to take into account all the possible situations that might occur. This task becomes harder for novice users as they are new in carrying out the design task. Thus, it is very likely that while designing behaviors, the users would forget to take into account certain situations. It would be helpful if the users were able to observe a simulation of the created behaviors to immediately identify any issues with them.

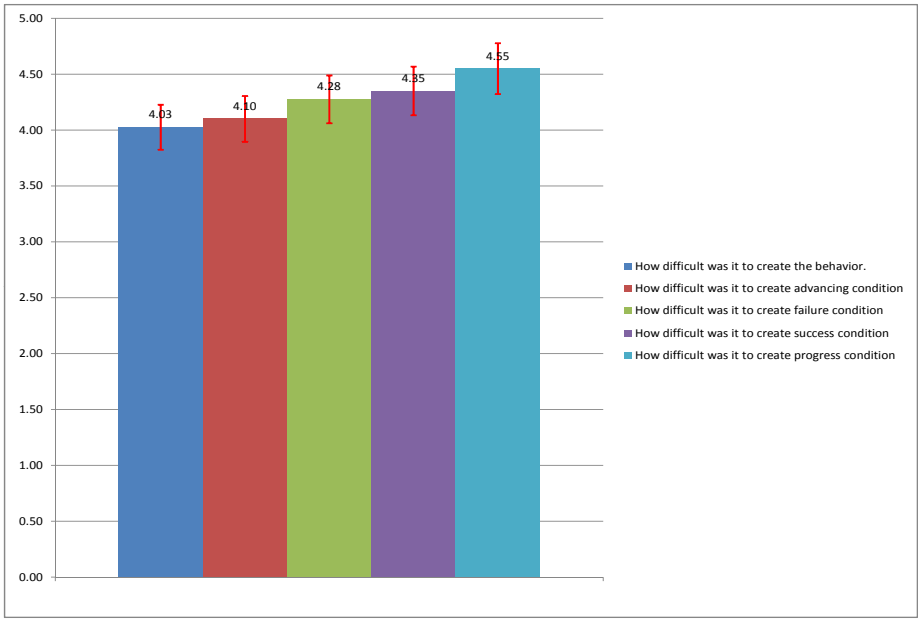



**Figure 6:** The figure shows the numerical ratings on various questions asked at the end of the study. 0-not difficult at all, 5-very difficult

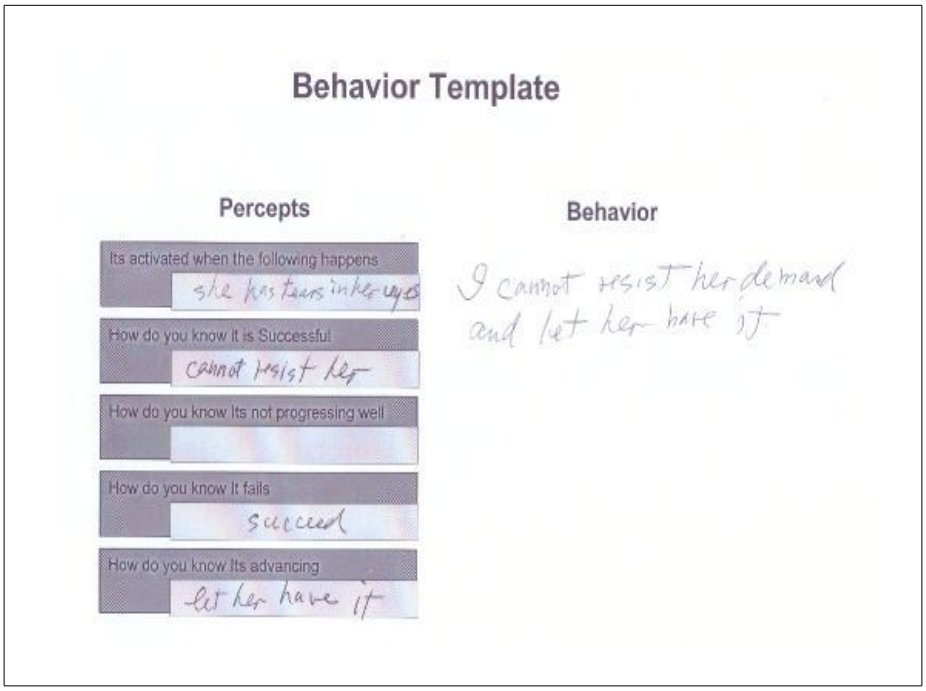



**Figure 7:** The figure shows a form related to behavior conditions filled by the user.

#### *3.2.3.3 Remove Difficult Portions of Behavior Conditions*

There were parts of the behavior condition creation process that were left empty by the user. Users had difficulty identifying things like the "progress-condition" (30 out of 40 users) and "failure-condition" (34 out of 40 users) and at times "advancing condition" (28 out of 40 users) and "success-condition" (26 out of 40 users). Figure 7 provides an example behavior condition form filled by the user where the above mentioned conditions have either not been filled or filled with very vague descriptions, showing that they had difficulty filling out these conditions. P24 mentioned :

*When my behavior needs to be activated was clear. I know when I want my character to perform something. Some of the other things were difficult like success condition. I may not be able to know whether my behavior is successful or not by looking at things in the environment.*

On the question of difficulty level in creating advancing condition, users provided an average rating of 4.1 (0-difficult and 5-very difficult) as shown in Figure 6. Similarly for difficulty level in creating failure, success and progress condition users provided an average rating of 4.28, 4.35 and 4.55 respectively as shown in Figure 6.

One of the ways the problem of creating all these conditions could be addressed would be to reduce the set of conditions. On the one hand, this takes away some of the expressivity of the system, on the other, this helps make the authoring process more accessible for novice users.

#### *3.2.3.4 Provide for Simplification of Behavior Condition Linking*

Users had difficulty linking behavior conditions with lowest level sensory information that was available in the environment as P32 mentioned:

*Sometimes I could say when my behavior fails. But in general I wasnt sure if I could see these things in terms of what that was available in the environment.*

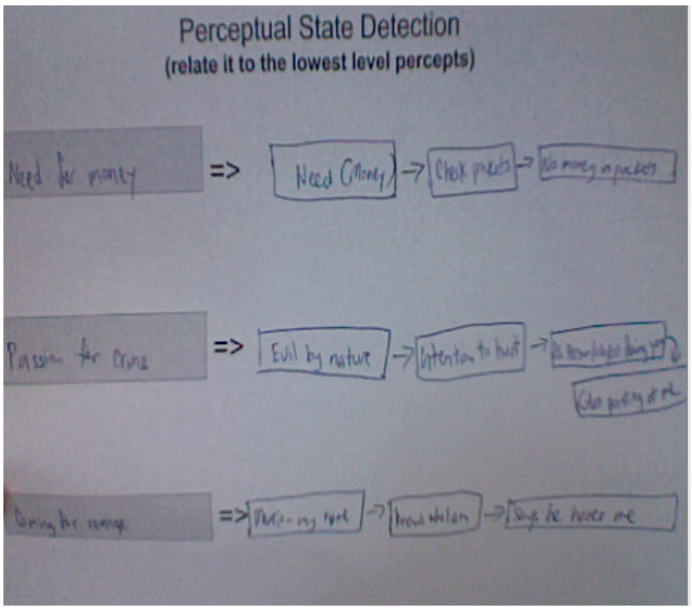



**Figure 8:** The figure shows an example behavior condition that users had difficulty connecting to lowest level sensory information

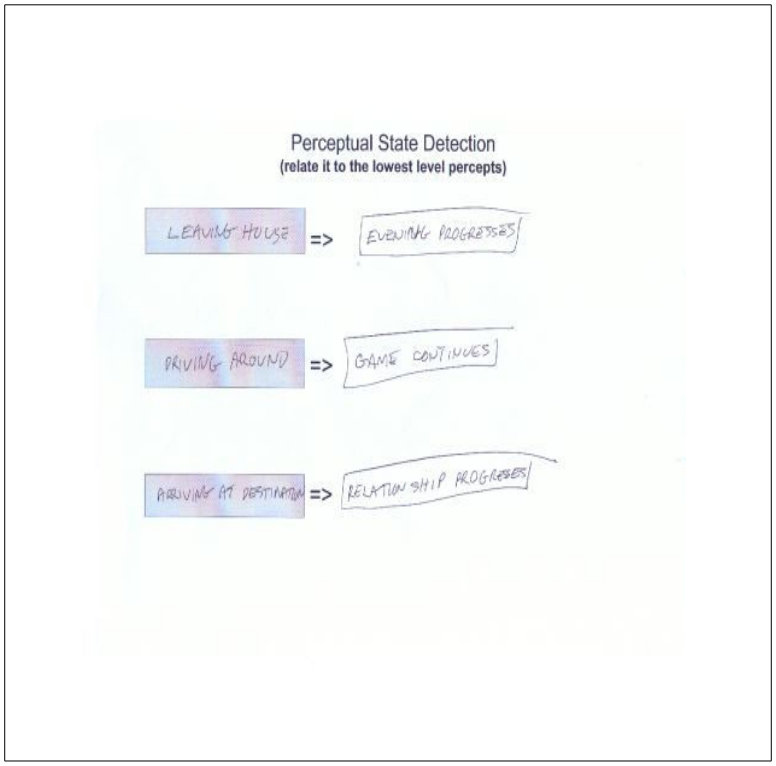



**Figure 9:** The figure shows another example behavior condition that users had difficulty connecting to lowest level sensory information

The perceptions at times could become very abstract and users had difficulty relating it to sensory information that can be detected from the environment. Figure 8 and 9 shows two such example conditions user had difficulty with. In Figure 8, "need for money" is broken down to the lowest level to "money in pocket". User had difficulty relating this to the lowest level with sensory information such as trying to describe "there not being money in someone's pockets". Apart from that, in order to concretely connect the sensory information obtained from the world to higher level behavior conditions, mathematical operators needed to be defined in certain situations and contexts (as shown in Figure 5). Most of the times users skipped this process. They felt that creating "mathematical formulas was the most difficult part"(as reported by subject P6) and they needed to have technical skills in order to carry out this step. For instance, subject P25 stated:

> *Defining it mathematically in terms of the lowest level things that are happening in the environment was unclear. I would think someone who has the technical skills would be able to understand it better than me (P25)*

One possibility would be to remove the notion of complex conditions defined using mathematical operators over the basic sensory information could be to provide a base set of parameterized conditions that can be used individually or combined together to make more complex conditions.

## *3.3 Chapter Summary*

In this chapter, I have looked at the first design pass over the representational vocabulary. I also presented the authoring activity that the novice user used to create behaviors. I have presented results from a user study conducted to evaluate the paper prototype version of the authoring activity. One of the interesting result that came out of the study was that novice users had difficulty conceptualizing the behavior

creation process as a hierarchical, top-down process. Briefly, the findings I presented in this chapter were:

- Authors had difficulty conceptualizing the behavior authoring task as a hierarchical top-down structure and were more attuned to thinking of the task in a sequential manner.
- In order to concretely see the effect of their creations, users expressed the need for concrete feedback on the authoring activity for them to be able to immediately identify problems with the created behaviors.
- As part of the behavior creation process, authors had difficulty creating many behavior conditions.
- The steps required to link these behavior conditions to lower level percepts needed to be simplified. The users had a lot of difficulty relating behavior conditions to lowest level perceptions.

Based on the results from the study, some of these insights were incorporated into the next version of the design.

In the next chapter, I present the current representational vocabulary and the interaction design approach used as part of the authoring process that also reflects the results inferred from the analysis of the first design prototype.

# CHAPTER IV

# AI BEHAVIOR AUTHORING

In the previous chapter, I presented the results from the paper prototype based user study. The user study provided some key insights and design directions for creating the authoring interface. The study indicated the following results:

- The need for having concrete feedback on the authoring activity so that the authors could see the results of the authoring process and therefore identify problems with it.
- The need to simplify steps required to link behavior conditions to lower level percepts.
- The need for simplification of behavior conditions as there were many behavior conditions that users had difficulty in creating.
- The insight that the authoring task should not require creation of deep hierarchical structures and should be presented as a sequential process.

In this chapter, I am going to present the design changes carried out to address these issues. This chapter presents in detail my answer to one part of the first research question i.e. the design solution that would enable novice users to successfully construct AI behaviors. Evaluating whether the novice users are able to "easily construct AI behaviors", the second part of the first research question is carried out later in Chapter 6. I discuss the current authoring interface and the activity that the users need to conduct. I also present the current architecture of the system (named Second Mind) detailing the behavior execution system and the interface layer that is used to connect it to external world.

## *4.1 Representation Vocabulary and Authoring Activity: Second Pass*

### 4.1.1 Simplifying behavior conditions

During the behavior condition creation step (step (c) mentioned in Section 3.1.2) of the user study, it was observed that some of the conditions were left empty by the user. In order to simplify things, some of these conditions have been removed in this version. The behavior condition that is now being used is the activation condition presented in Chapter 3 and is named "triggers". Reducing the set of conditions from the ones in Chapter 3 probably takes away from some of the expressivity of the system, however, this would allow keeping the authoring process simpler than the previous version. The conditions associated with the behaviors were therefore the following:

- **Triggers**: These are conditions that help the avatar identify when the behavior needs to be activated and carried out. An example would be a *trigger* with value "close to other avatar(any-avatar)" associated with the greeting behavior.

Another difficulty that was observed during the study was the linking of behavior conditions with lowest level sensory information. The conditions at times, could become very abstract and users had difficulty relating them using mathematical operators to the sensory information that can be detected from the environment. As a result, notion of complex triggers that can be defined using mathematical operators over the basic sensory information was removed. A base set of triggers (described next) were provided that could be used individually or combined together to make more complex triggers.

The following set of triggers were defined:

- **User Clicks on (button_label)** : This trigger is used when the user wants the avatar to perform a behavior in response to the player (who is interacting

with the avatar) clicking a button. The value of "button_label is provided by the author during the authoring process.

- **Avatar Close to (object_name)** : This trigger is used when the author wants the avatar to perform a behavior when the avatar is close to an object with the name "object_name" in the current map. Users can select the object name through a drop down list. All possible objects close to the avatar are dynamically changed depending upon the current location of the avatar.

- **Avatar Touches (object_name)** : This trigger is similar to Avatar close to (object_name) except the trigger is used when the user wants the avatar to perform a behavior when the avatar touches an object with the name "object_name"

- **Player Close to (object_name)** : This trigger is used when the author wants the avatar to perform a behavior when the player (who is interacting with the avatar) is close to an object with the name "object_name"

- **Player Touches (object_name)** : This trigger is used when the author wants the avatar to perform a behavior when the player touches an object with the name "object_name"

- **Behavior Finishes (behavior_name)** : This trigger is used when the author wants the avatar to perform a behavior immediately after another of avatar's behavior with the name "behavior_name" finishes.

- **Silence after behavior finishes (behavior_name, Xsec)** : This trigger is used when the author wants the avatar to perform a behavior when another of avatar's behavior with the name "behavior_name" finishes and there is silence for a time interval of duration "Xsec" after finishing of the behavior.

- **Event happens (event_name)** : This trigger is used when the author wants the avatar to perform a behavior when a particular event happens in the virtual world environment. This trigger is used to communicate with external entities in the environment like other active objects. The objects can send a message with event_name through an http call to the Second Mind server.

- **And trigger(A, B)** : This trigger is used when the author wants the avatar to perform a behavior when both of the triggers A and B happen. A and B can be any of the triggers defined earlier.

- **Or trigger(A, B)** : This trigger is used when the author wants the avatar to perform a behavior when either of the triggers A or B happen. A and B can be any of the triggers defined earlier.

In order to present the authoring task, a interaction design approach was selected to make it more intuitive for the users to author the behaviors. I briefly discuss them next.

### 4.1.2 User Interaction approach employed

One of the results from the user study was that users had difficulty conceptualizing hierarchical behavior structures. One of the design decisions that had to be made was to remove the notion of deep hierarchies. Based on the user feedback that they were more attuned to thinking of the behavior creation process as a sequential process, the notion of timelines was used. Timelines are employed by most popular professional video editing packages. Timelines indicate a left-to-right sequential sequence, where each event occurs in chronological order. The construction of behavior would thus involve creating a left-to-right sequence represented on a timeline. Figure 10 shows the construction of a greet behavior. It consists of a basic actions like "salute" and two other conversational actions represented in a left-to-right sequence on the timeline. In

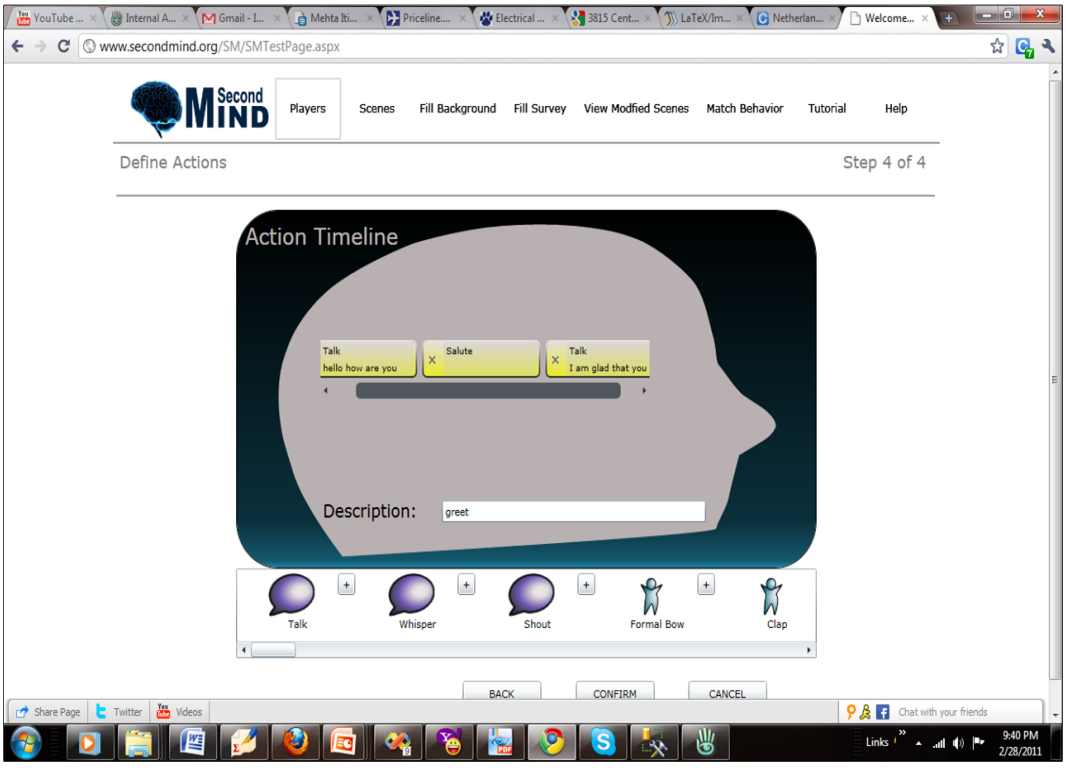



**Figure 10:** The figure shows the screenshot of the behavior authoring process

order to help the user conceptualize the task more concretely, the authoring activity is presented as designing the mind of the agent. The timeline is presented in a mind-like shape (shown in Figure 10) to reflect this idea. All the behaviors are shown in a sequence using the coverflow approach[1] (shown in Figure 11).

Timelines have been employed successfully as an interaction design approach in various storyboarding authoring tools like Adobe's Director and Apple's Garageband (shown in figure 12). Timelines have been also used by AI based interactive story authoring tools like Scribe[72], U-Create[95] and Viper[22] (shown in Figure 13 (A),(B) and (C) respectively). The timeline is used to represent the ordering of the story segments in a left to right sequence. The story segments can be reordered on the timeline as well. A hierarchical approach for representing the story segments has

[1]Cover Flow is an animated, three dimensional graphical user interface that has been integrated within iTunes and other Apple Inc. products for visually flipping through snapshots of documents, website bookmarks, album artwork, or photographs. (Definition taken from wikipedia)

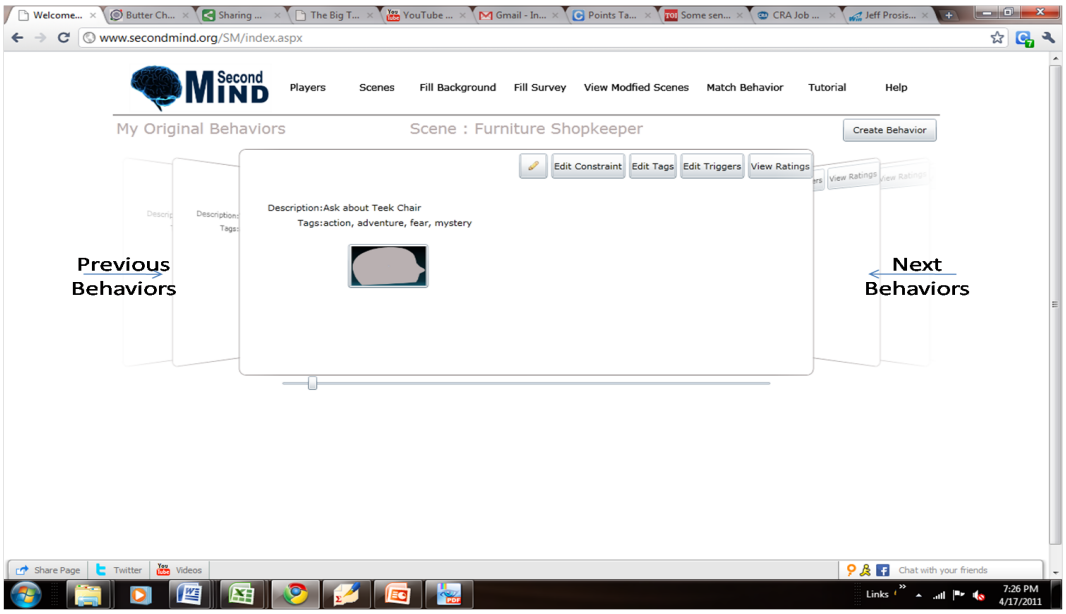



**Figure 11:** The figure shows the behaviors for a scene shown using coverflow.

been employed in Storycanvas (shown in figure 14). As demonstrated in Chapter 3, a hierarchical goal-subgoal representation of the behavior authoring process is difficult for the novice authors to understand. For representation of the behavior construction process, timelines provide a more natural way of presenting the behavior construction process, in a left to right sequence. There are two ways in which the timeline based interaction design approach used in Second Mind differs from the existing timeline based AI story authoring approaches:

- I provide empirical evidence that a left to right organization (represented by a timeline based interaction design approach) represents a more natural way for novices to think about the behavior construction process. Chapter 3 results indicate that when novice users were visualizing the creation process, it was very hard for them to think in a top down fashion and they were more used to thinking of things in a sequential fashion.

- I provide empirical evidence (in Chapter 6) that timeline based interaction design approach can be used successfully by novice authors to create AI behaviors

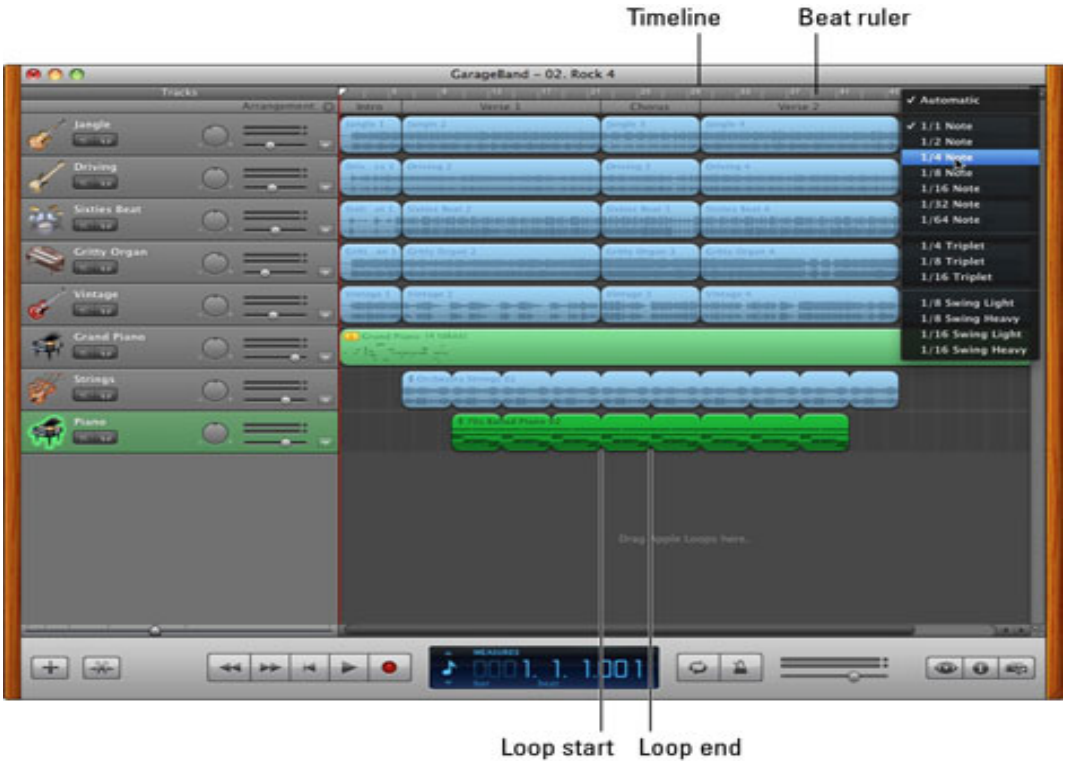



**Figure 12:** The figure shows the screenshot of Apple's Garageband timeline based storyboarding process

in an easy and understandable manner.

- The timeline based interaction design approach used in Second Mind is used for presenting AI behavior construction instead of presenting the timing information for story segments.

Another result from the user study was the need to provide more immediate and concrete feedback to the user in order to view the results of the created behaviors. Let's briefly look at the steps taken to solve the issue.

### 4.1.3 Concrete Feedback

In order to provide concrete feedback to the user, the authoring environment is connected to a virtual world named Second Life [1]. Second Life is selected as the virtual environment in which agents behavior would be executed. Second Life is a virtual

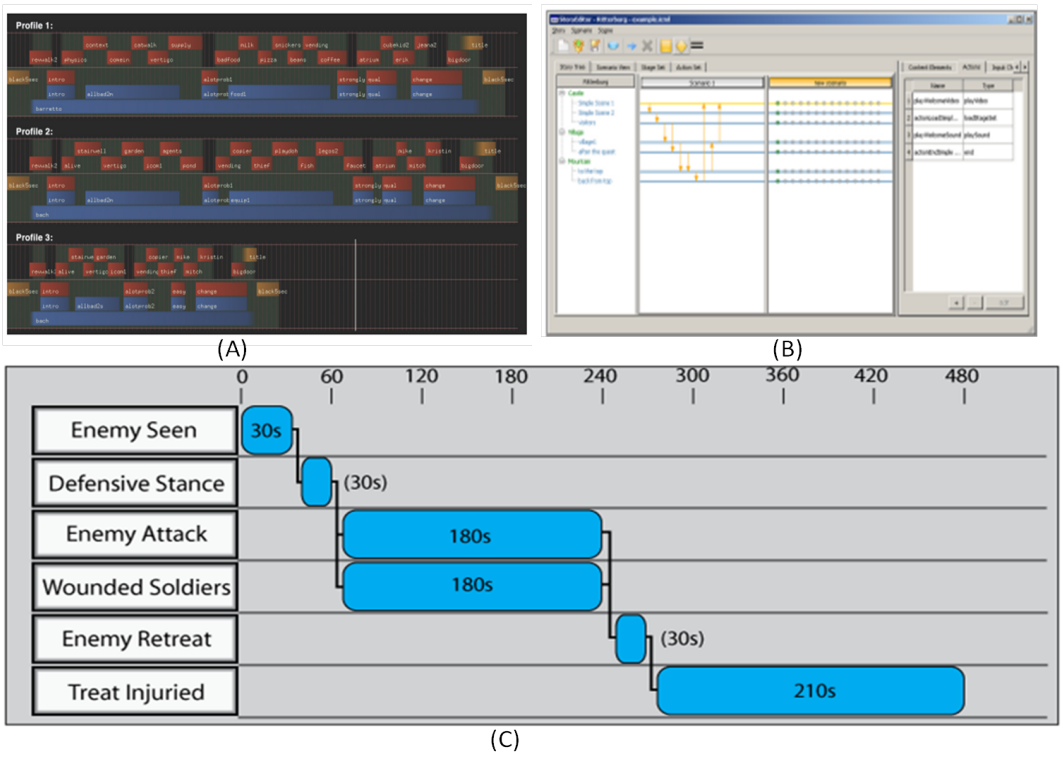



**Figure 13:** The figure shows the screenshots of timeline based interaction design approach used by interactive story authoring approaches

world that allows registered users to exist in a rapidly developing world. Users create avatars, or virtual representations of themselves that allow them to maneuver and physically exist in the virtual world. Second Life allows users to create, design and build their own clothes, buildings, artifacts, landscape, pets and much more. There were three reasons for picking Second Life as the environment. First, there is a large user base that are present inside Second Life. Second, it allows for easy connection to external controllers through an API. Third, it provides a rich set of basic actions (in the form of emotions and physical actions) [2] that can be executed by the avatar. Through the authoring interface, users is able to create behaviors and execute them in Second Life at any time in order to concretely see the effects of their authored artifacts. Seeing the results of their authored behaviors would allow the users to carry out necessary modifications and find any issues with them.

Next, Let's look at the authoring activity that the user carries out.

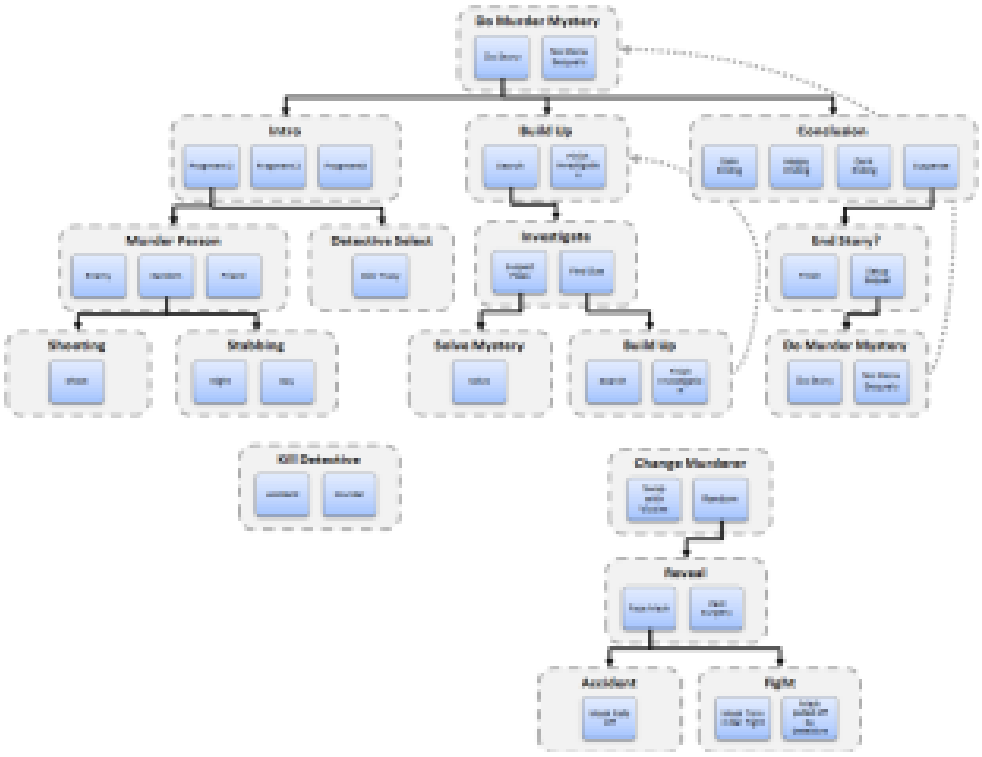


**Figure 14:** The figure shows the screenshot of hierarchical story representation in storycanvas

## *4.2 Authoring Activity*

### 4.2.1 Step I. Account Creation

In order to use Second Mind, the user first needs to create an account on the Second Mind website[2]. The account creation form is shown in Figure 15. He is asked to enter his username, email address and password in order to create the account. Once the account is created, the user goes through a guided tutorial which explains him the various terminologies used in the authoring interface (like scenes, behaviors, actions, triggers etc). The user is also given details about the steps he need to undertake in order to connect his Second Life avatar to Second Mind (described later in section 4.3.2). He then creates his first player by entering information in the screen shown in Figure 16. The user is asked to enter information like player's name, description and

[2]http://www.secondmind.org/


**Figure 15:** The figure shows the screen for creating the account in Second Mind

Second Life login credentials among other things. User can create multiple players and multiple Second Life avatars to give his players different physical manifestation. Once these initial steps are performed the user can start creating his first scene and corresponding behaviors.

### 4.2.2 Step II. Scene Creation

As the next step, user creates a new scene. The scene provides a social context encompassing all the behaviors within that scene. An example of a scene is a furniture shopkeeper where the avatar acts as a shopkeeper selling furniture items. User first creates a description of the scene by entering information like the name and its description and selects a player created in earlier steps (shown in step Ia in Figure 17). The user also associate constraints at this step. Currently, there are two constraints namely a) likeable and b) good salesman that the author can associate with a scene. By default, all the scenes are associated with two constraints a) overall experience

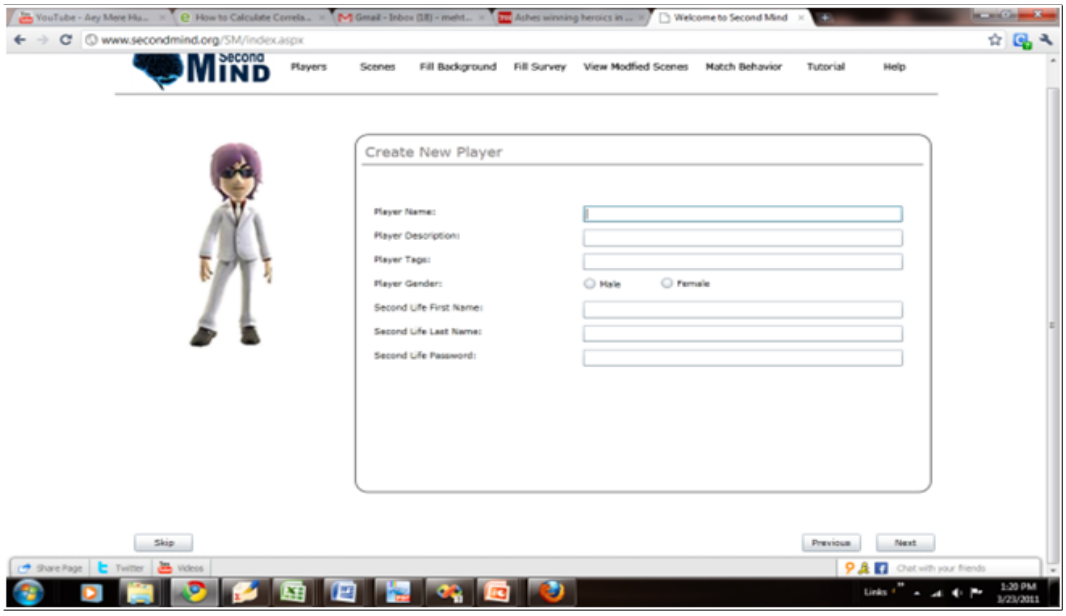



**Figure 16:** The figure shows the screen for creating the player in Second Mind

and b) avatar's performance. These constraints provide a way to measure the success of the overall scene. The author uses a set of sliders to specify values (on a 0-5 point scale) for the constraints. User also associates tags with the scene during this step. For example, he could associate a tag like "furniture" with the shopkeeper scene. In step 1b, the user adds existing behaviors associated with the player that was selected in step Ia. The scene would now have a copy of these existing behaviors that could be used as a starting point. In step Ic, user confirms his selection and he can now see the created scene (shown in step Id).

Once the scene is created, user starts creating behaviors for the scene.

### 4.2.3 Step III. Behavior Creation

The user first creates a behavior by entering information like the name and associated constraints for the behavior. Each behavior is associated with constraints that provide a way to measure the success of the behavior sets. The user uses a set of sliders (shown in Figure 18) to specify values on a 5 point scale for the constraints. For example, the author specifies that he wants the shopkeeper to be highly enthusiastic (setting a

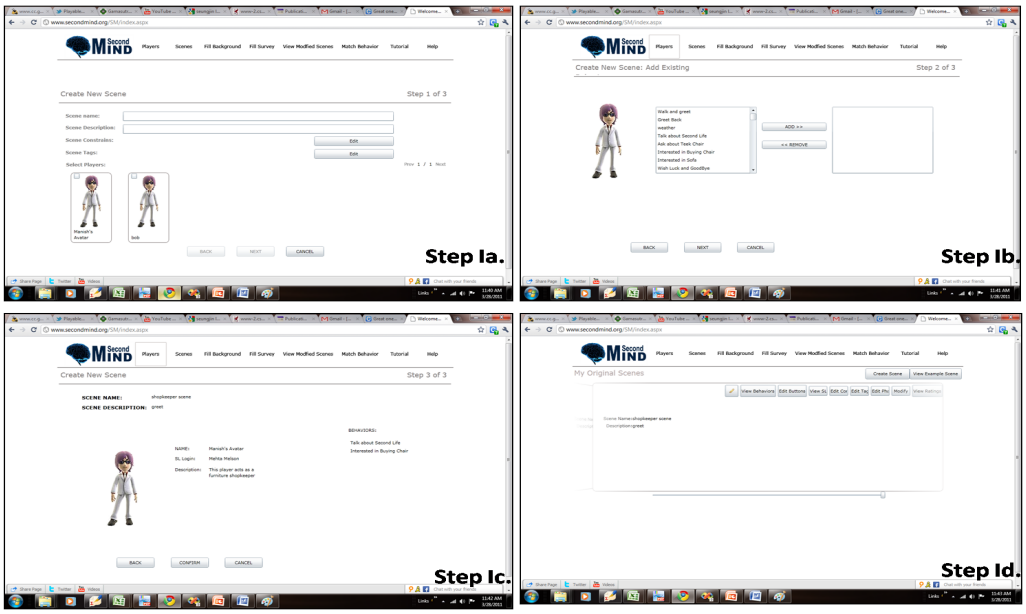



**Figure 17:** The figure shows the screen for creating the scene in Second Mind

high value like 5 on the slider) and less pushy (setting a low value like 0 or 1 on the slider) while performing the greet behavior. At the end of the player interaction with the avatar, the player is asked to provide feedback on individual behaviors based on the defined constraints. The constraints will be described in more detail in Chapter 5. User also associates tags with the behaviors during this step. For example, user would associate a tag 'greet' with the behavior "greeting the customer".

Next, the user is presenting with a list of recommended behaviors that he can use as a starting point (as shown in Figure 20). The user can either pick a recommended behavior or ignore the provided suggestions.

The user then adds/modifies triggers for the behavior by selecting them from the list of available triggers described earlier in section 4.1.1. As described earlier, triggers help the avatar identify when the behavior needs to be activated and carried out. An example of a trigger associated with the greet behavior is shown in Figure 21. The trigger is combination of "Close to other avatar(any avatar)" or "User Clicks on Button(Greet)".

The user then adds/modifies the basic actions that the avatar will perform when

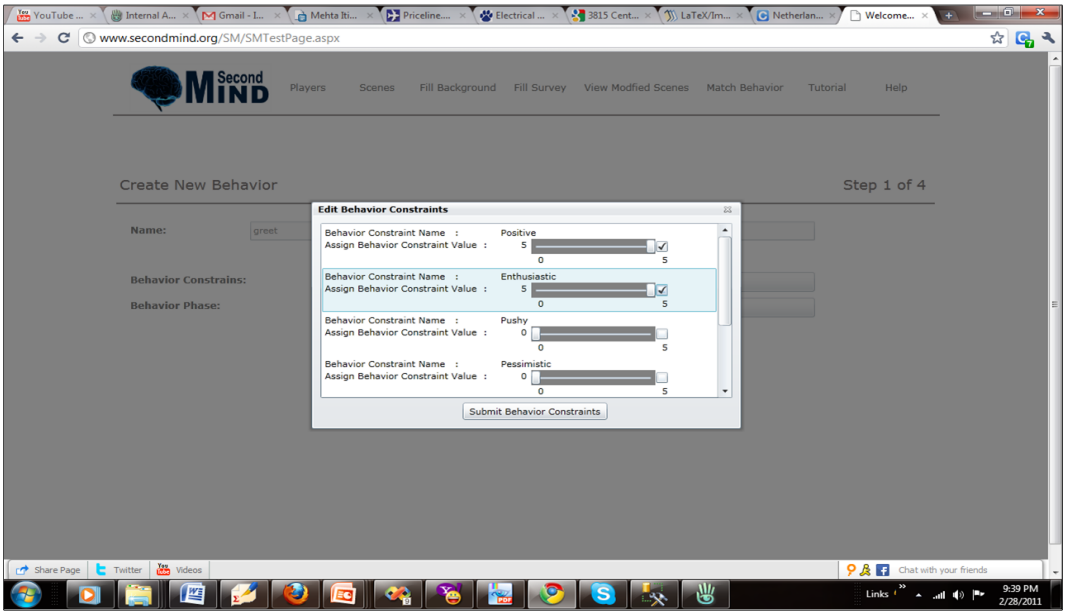



**Figure 18:** The figure shows the screenshot of the UI for defining the constraints

the behavior is triggered. Figure 10 shows three actions added as part of the greet behavior. It consists of a talk action with parameter "hello how are you", salute gesture and talk action with parameter "I am glad to see you here". The complete list of actions is available at second life website[2].

The user can iteratively repeat step III and define new behaviors for the scene. Once the behaviors are created, user can login in Second Life and test the behaviors.

### 4.2.4 Step IV. Second Life Interaction

In order to interact with an avatar in Second Life, a user in Second Life walks to the avatar and click on an attachment on his left shoulder and is presented with a user interface (shown in Figure 23). The user is presented with a list of scenes that have been authored for the avatar and selects a scene to interact with the avatar. Once the scene has been selected, he is presented with a list of buttons corresponding to the selected scene. These button labels are loaded from the Second Mind database based on the selected scene as shown in Figure 22. These button labels are based on various "User Clicks on (button_label)" triggers associated with the behaviors of

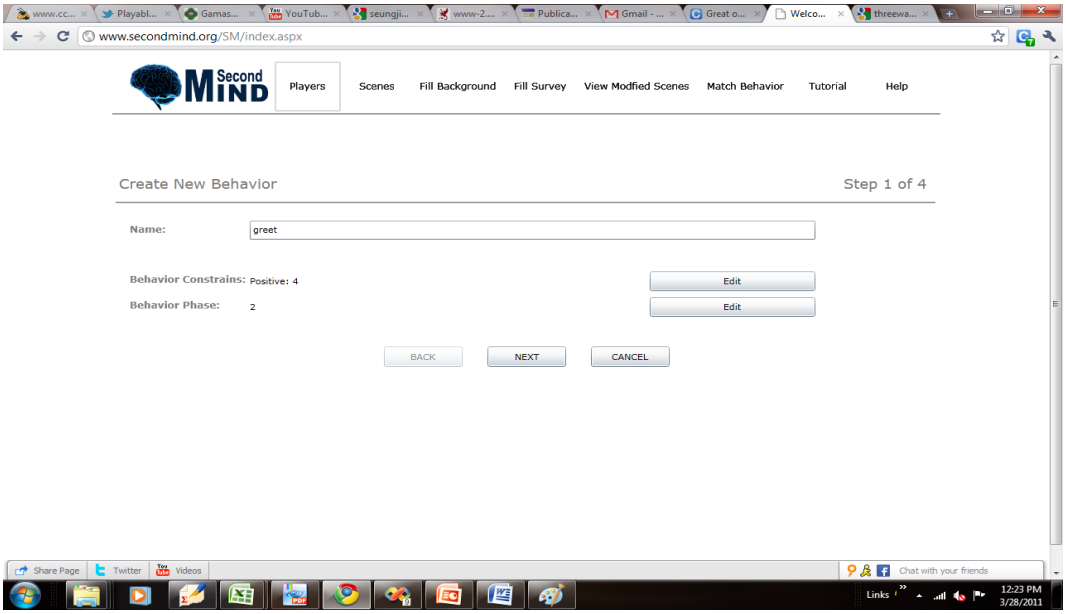



**Figure 19:** The figure shows the first screen for creating the behavior in Second Mind

a scene. The user clicks on the buttons to interact with the avatar. Avatar will also take initiative by carrying out some behaviors on its own. Avatar initiative would depend upon if the author has defined behaviors that the avatar could initiate on its own and whether the current world state satisfies the triggers defined for these avatar initiated behaviors.

Once the interaction finishes, user is presented with a list of questionnaire to gather feedback on avatar's overall performance and individual behaviors as shown in Figure 25. User is presented with a feedback form where he is asked quantitative questionnaire. The questionnaires are automatically generated using template questions that are targeted towards eliciting information on the constraints associated with the behaviors. For example, a template used for generating the question is *"Was the character "XX" when he performed "YY" behavior"* where XX is replaced by the name of the constraint and YY by the name of the behavior. So a generated question on positive constraint associated with a behavior named "need a couch" is *"Was the character "positive" when he performed "need a couch" behavior"* where the values XX and YY have been replaced. The user provides the feedback (on a 5 point

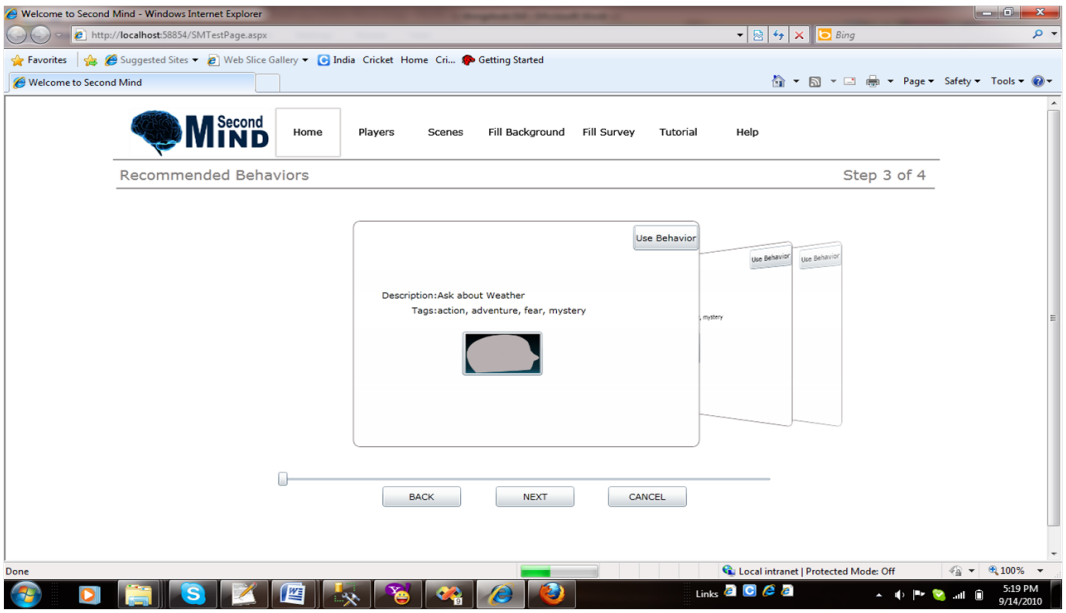



**Figure 20:** The figure shows the recommended behavior

scale) on the behaviors by answering the questionnaires on associated constraints. The user can also look at the details of the behaviors (essentially the actions) that were executed . The user also provides feedback on overall scene performance. This feedback is then stored along with the execution trace of his interaction in the Second Mind database.

With a basic understanding of the authoring activity, lets look at the system architecture.

## *4.3 System Architecture*

Second Mind architecture (shown in Figure 26) has four core components and one application-specific component. The four core components are:

- **Second Mind Database:** a MS-SQL server based database that stores scenes, players, behavior elements such as triggers, basic action and interactive responses. The various tables that are part of this database are shown in Figure 75 in the Appendix.

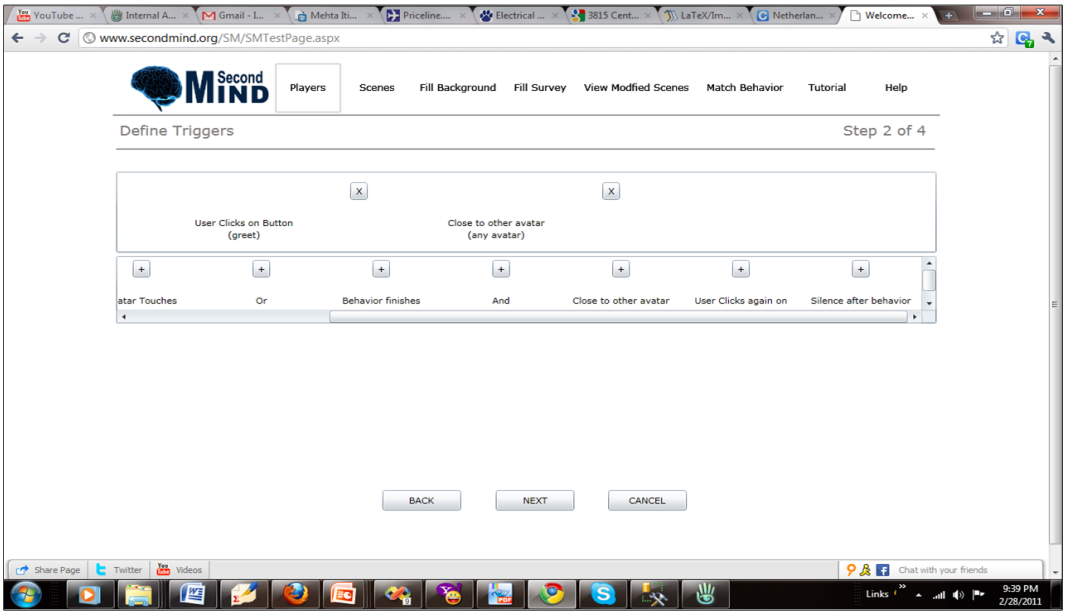



**Figure 21:** The figure shows the triggers defined for the greet behavior

- **Behavior Execution Engine:** an execution engine that loads the behavior elements and executes them in the external world. The behavior execution engine connects to the external world through the interface layer that is specific to the world that Second Mind is connecting to.

- **Web Interfaces:** provides the interfaces for creating user account, scenes and behaviors as described earlier.

- **Behavior Recommendation Engine:** provides suggestions during authoring on behaviors that can be used as a starting point during the behavior creation process.

In order to apply Second Mind to a specific application, one application-specific components is required:

- **Interface Layer**: The architecture of Second Mind is domain-independent and can be connected to different virtual world simulations via a custom interface layer. The interface layer provides the ability to connect Second Mind to the

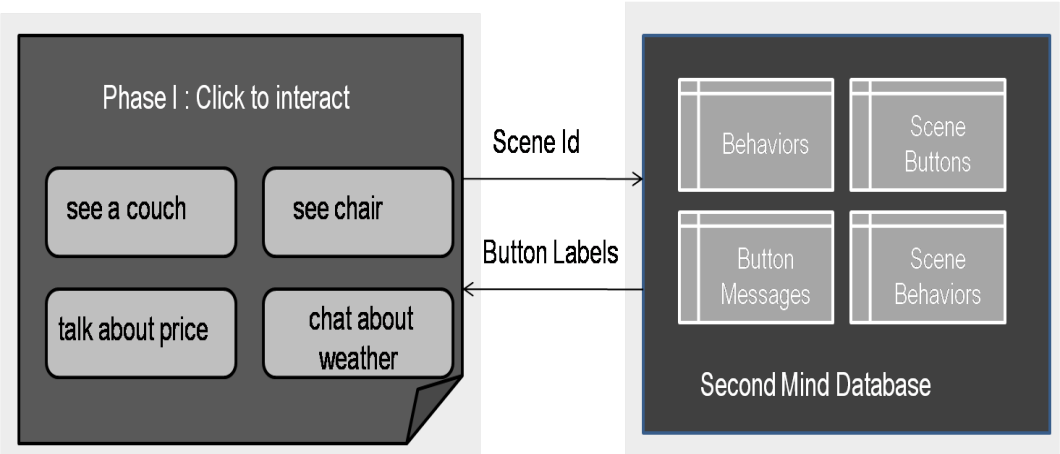



**Figure 22:** The figure shown the process of creating second life interface buttons that are presented to the player during his interaction

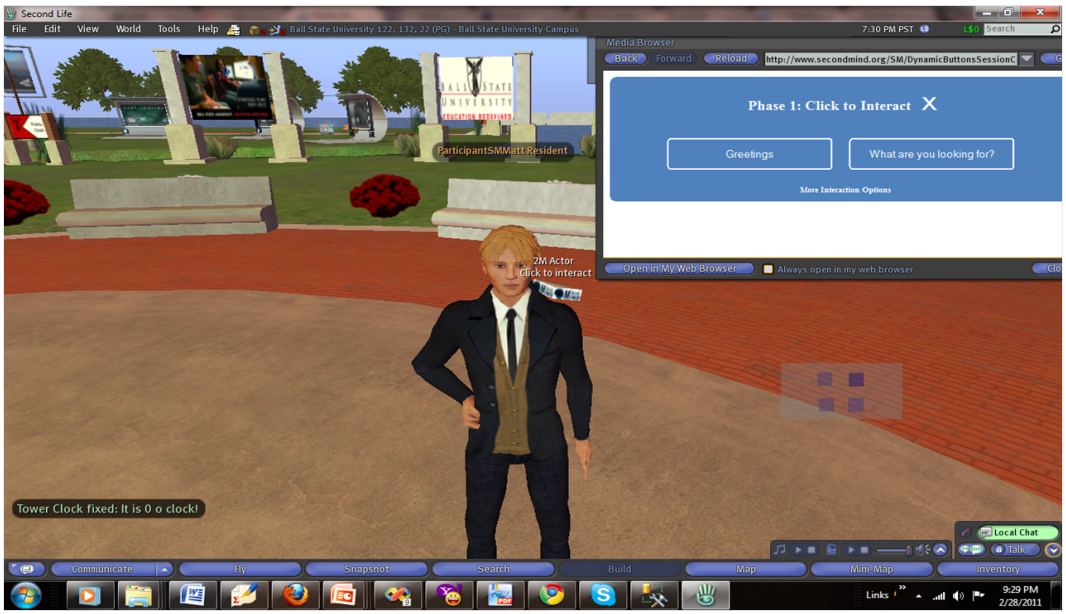



**Figure 23:** The figure shown a screenshot of the player interaction in Second Life.

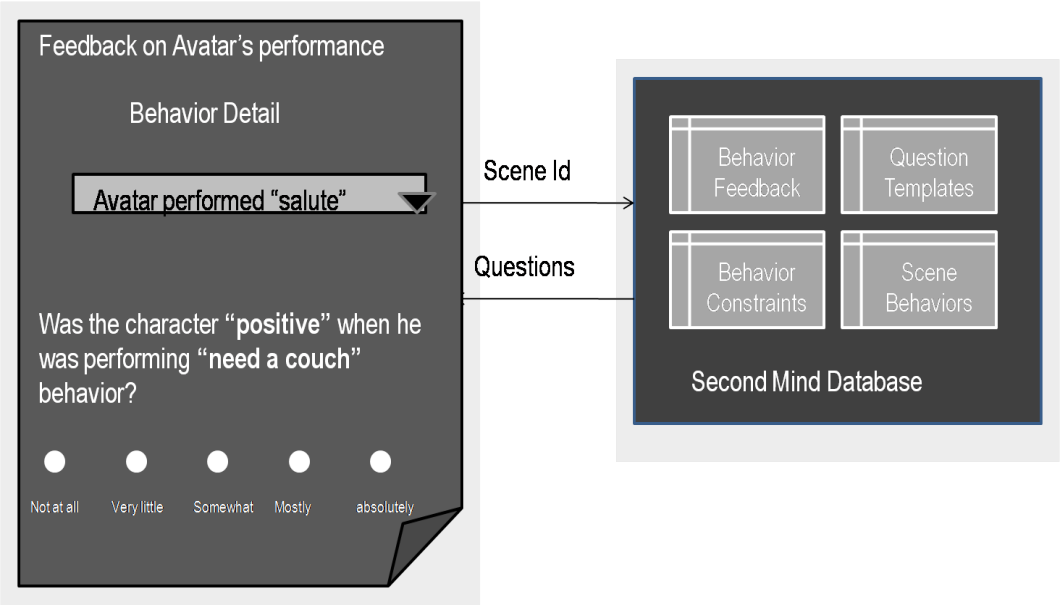



**Figure 24:** The figure shows the question generation process for getting feedback on avatar's performance from the player at the end of his interaction


**Figure 25:** The figure shows the questionnaire interface in second life that is presented to the players after their interaction with the avatar in second life

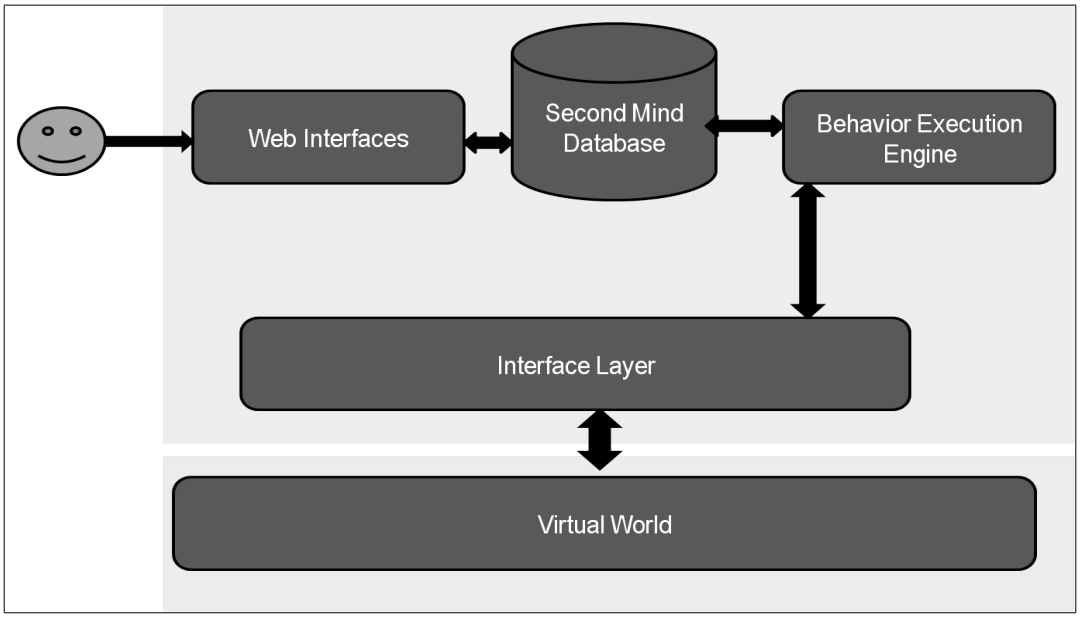



**Figure 26:** The figure shows second mind architecture

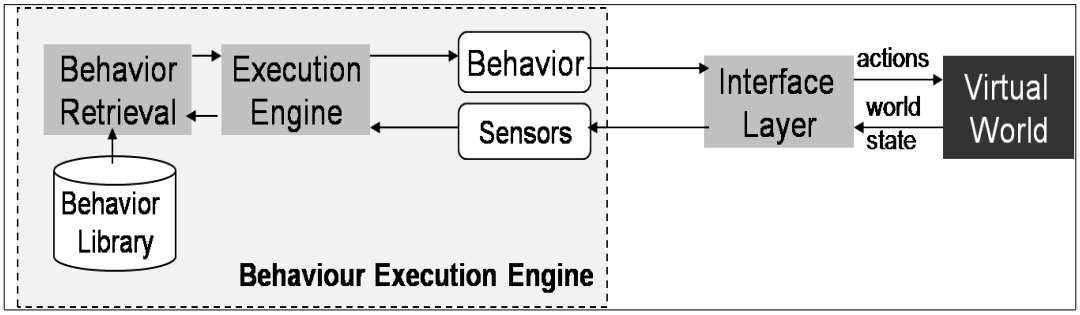



**Figure 27:** The figure shows details of the behavior execution engine

simulation platform, such as Second Life or other virtual world. The interface layer requires definition of percepts, actions, and other application-specific elements along with APIs to connect Second Mind to the simulation. More details about the connection to Second Life can be found later on.

Let's look at some of the components in more detail.

### 4.3.1 Behavior Execution Engine

The behavior execution engine loads the behaviors corresponding to the currently active scene i.e. the scene that the user has selected to interact with the avatar,

identifies the active behavior(s) based on the current perceptual information from the world and sends the actions corresponding to active behavior to the world for execution through the interface layer. In order to accomplish this, the behavior execution engine consists of the following modules.

- **Behavior Library:** part of the Second Mind Database that holds behavior elements for a specific application domain, such as a Second Life avatar or a household robot. For example, if the domain of application is education, the library may contain personalities corresponding to teacher, student, etc. Similarly, for a household robot, the library may contain personality elements corresponding to greeting guests, cooking food, or cleaning the house in different ways.

- **Behavior Retrieval:** loads all the behaviors corresponding to the active scene from the behavior library. The active scene is selected by the player in Second Life, when he is presented with all the possible authored scenes corresponding to the interacting avatar.

- **Execution Engine:** receives the currently active domain independent sensors from the interface layer and checks triggers for the behaviors corresponding to the active scene to see if any of the behaviors have become activated. Every cycle it also takes the action corresponding to the current active behavior and sends it through the interface layer to the virtual world for the avatar to perform it .

Next, lets look at the interface in more detail.

### 4.3.2 Interface layer

Interface layer parses the domain dependent perceptual state into a set of domain independent sensors (shown in Figure 29). The domain dependent perceptual state

from Second Life contains the following things:

- **Object Info:** contains information regarding various objects in the environment, their current position, rotation, type (whether an avatar or a passive object) and the current region to which the object belongs. The region information helps detect whether the object is in the same region as the avatar or not.

- **User Button Click Info:** contains information regarding button clicks made the by the user during his interaction with the avatar.

- **Event Info:** contains information regarding various events happening in Second Life. These events are, for example, things like user moving around in the environment or touching an object. Other objects in the environment can send various events to communicate with the avatar. If there are behaviors that have "Event happens (event_name)" as triggers, avatar can react to events with name "event_name".

The world specific information is parsed into a set of domain independent sensors. These domain independent sensors shown in Figure 29 correspond to the various triggers that have been described earlier. This information is then sent to the execution engine which checks the sensors against the triggers corresponding to the current set of behaviors to identify if any new behavior(s) has become activated. The interface layer module also receives action corresponding to the active behavior and sends it to the virtual world. The interface layer provides an abstraction layer and maps the action name received from the execution engine to virtual world specific action name. For example, action laugh is mapped to action express_laugh which is the specific name used by Second Life virtual world. Table 2 shows some of the mappings for the virtual world Second Life.

In order to connect to Second Life, at first, the open source version of Second Life client[3] was used and modified it to connect with Second Mind to a) send the perceptual information from Second Life to the interface layer and sending the actions from the interface layer to Second Life to control the avatar. This, however required users to download a modified Second Life client which would have become a tedious process for users. As a result I decided to use the existing Second Life client and created scripts in Second Life's proprietary scripting language named Linden Scripting Language (LSL)[4] for sending the perceptual information from Second Life to the interface layer and sending the actions from the interface layer to Second Life for controlling the avatar. The scripts are shown in Figure 28. The following four scripts were created:

- **Animation Controller**: This script receives the messages from the animation server script and performs the necessary steps in the form of calling appropriate LSL commands to have the avatar perform the actions that are sent across from the behavior execution engine through the interface layer.

- **Percept Listener**: This script collects the world information from Second Life by calling appropriate LSL commands and sends it in an appropriate format to the perception server script.

- **Animation Server**: This script contains a web server that is listening for action messages from the interface layer. In order to send the actions to this web server, interface layer retrieves its web address from the Second Mind Database and makes an http call to send the actions. The animation server then sends this action to the animation controller script.

- **Perception Server**: This script contains a web server that is listening for requests from interface layer to send across perceptual information. In order to

[3]http://wiki.secondlife.com/wiki/Source_downloads
[4]http://wiki.secondlife.com/wiki/LSL_Tutorial

**Table 2:** The table shows the action names used in Second Mind with the corresponding virtual world action name in Second Life

| **Domain Independent Action Name** | **Virtual World Action Name** |
|---|---|
| Laugh | express_laugh |
| Shrug | express_shrug_emote |
| Bow | avatar_hold_bow |

make a request to the perception server, interface layer retrieves the web address from the Second Mind Database and makes http calls to send the perception requests to the perception server. Once the request is received, the perception server returns back the perceptual information (received from percept listener) and sends it back to the interface layer.

The collection of scripts described above are packaged in a component called Second Mind Controller and distributed through the Second Life marketplace [5]. Second Life marketplace is an official web portal from Second Life where all the items from various vendors are listed and available for purchase. Second Mind controller is listed on the Second Life marketplace and is available for free. In order to have their avatar controllable through Second Mind, users need to download it from the Second Life marketplace. Purchasing items from Second Life marketplace is a common activity performed by Second Life users. Once users have bought the items from Second Life marketplace, Second Mind controller is downloaded automatically onto their avatars which they can then attach it to their avatar in Second Life. As the Second Mind controller is attached to the avatar, the animation server and perception server register their URLs within the Second Mind database by making an http call.

[5]https://marketplace.secondlife.com/

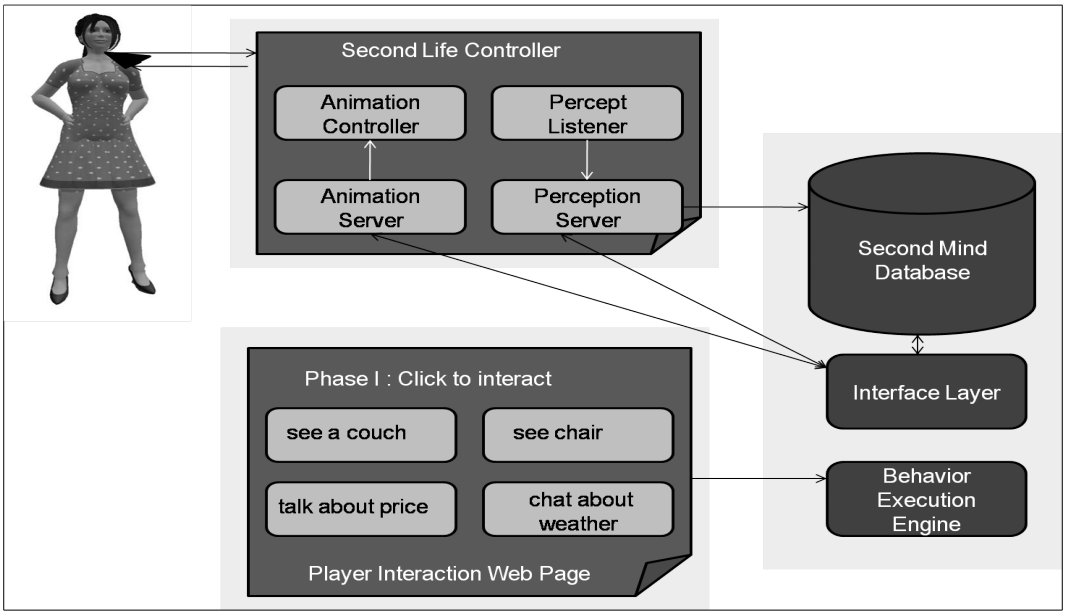



**Figure 28:** The figure shows details of second mind connection with second life through various scripts

### 4.3.3 Behavior Recommendation

During the behavior authoring process, once the name of the behavior is entered (in step III of subsection 4.2.3), the user is presented with a list (currently three) of recommended behaviors (shown in Figure 20) that he can use as a starting point. The behaviors are suggested based on similarity between the behavior name specified for the currently authored behavior and existing behaviors in the database. The semantic similarity is calculated based on a WordNet[6] based algorithm that is available here[7]. The algorithm for ease of reference is also described below:

Let's take an example of two behaviors to understand the algorithm. Let's say, the names of the two behaviors are sentences A and B respectively and lets say that the length of A is m and length of B is n.

I. In the first step, the sentences A and B are broken down into a list of tokens. Each sentence is partitioned into a list of words, and the stop words (like a, the

[6]http://wordnet.princeton.edu/
[7]http://www.codeproject.com/KB/string/semanticsimilaritywordnet.aspx

etc) are removed.

II. Next, the words of each of the sentences are stemmed by removing the common morphological and inflexional endings of words. For examples, word "tables" would have been converted to "table" at this step.

III. The next step is to identify the correct part of speech (POS)) (like noun, verb, pronoun, adverb) of each word in the sentence. The output at this step is a single best POS tag for each word in the sentence.

IV. The next step is to identify the most appropriate sense for every word in a sentence. At this step, dictionary definitions (called gloss) is used to disambiguate a word in a sentence context. In order to disambiguate a word, the gloss of each of its senses is compared to the glosses of every other word in a phrase. A word is assigned to the sense whose gloss shares the largest number of words in common with the glosses of the other words.

V. A semantic similarity matrix is built say, S[m,n] of each pair of word senses, where S[i, j] is the semantic similarity between the most appropriate sense of word at position i of A and the most appropriate sense of word at position j of B. S[i,j] is the path length from i to j and is measured using the distance between the two words i and j in the WordNet taxonomy structure.

VI. Semantic similarity between sentences is now the problem of computing a maximum total matching weight of a bipartite graph, where A and B are two sets of disjoint nodes. The algorithm that is used to find the semantic similarity is the Hungarian method for matching bipartite graphs[8].

VII. Finally, the similarity of the sentences is computed based on the similarity of

[8]Mordecai J. Golin, Bipartite Matching and the Hungarian Method, Course Notes, Hong Kong University of Science and Technology

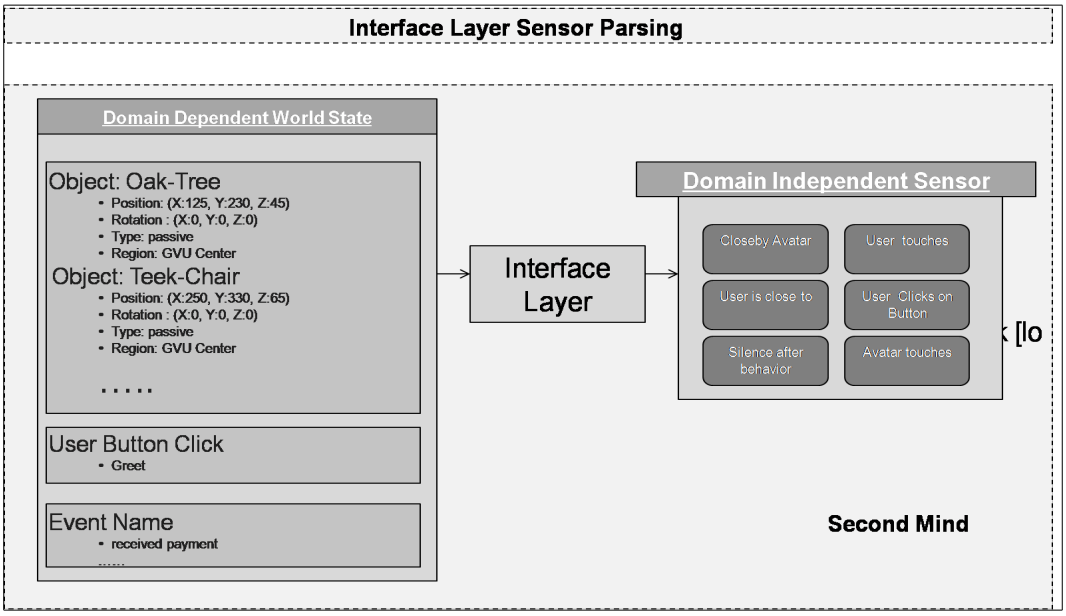



**Figure 29:** Interface Layer that parses the world state in Second Life into domain independent sensors

the pairs of words. In order to compute the similarity, the formulae that is used is 2*Match(x, y)/(m+n) where match(A, B) are the matching word tokens between A and B. So for example, if there are two sentences A and B and A and B have lengths of 3 and 2, respectively. The bipartite matching returns that A[1] has matched B[1] with a score of 0.6, A[2] has matched B[2] with a score of 0.5. Therefore the matching score will be 2*(0.6 + 0.5) / (3 + 2) = 0.44.

Similarity between the currently authored behavior name and the names of the behaviors is computed using the algorithm described above and the top three results are presented to the user (shown in Figure 20).

### 4.3.4 Technical Details

The authoring environment is available on the web[9]. It has been developed in Silverlight which is an application framework developed by Microsoft for writing and

[9]http://www.secondmind.org

running rich internet applications. The run-time environment for Silverlight is available as a plug-in for most web browsers. Silverlight was chosen as it provided us the ability to build graphically rich behavior editing environment. Microsoft has developed Silverlight and provides tight integration with all the Microsoft tools including Database (MS-SQL Server)integration and ASP .Net. Silverlight also provides a host of design tools like Microsoft Expression Blend to design user interfaces in an interactive, WYSIWYG environment. MS-SQL server was used for the Second Mind database. All these factors coupled with the fact that all the Microsoft tools were freely available through MSDN Georgia tech alliance decided us to choose these tools. The authoring environment was made web-based to keep it generic so that there was no separate download for the authoring environment and therefore could be accessed by the users easily.

## *4.4 Chapter Summary*

In this chapter, I have looked at the design changes carried on the representational vocabulary and the authoring activity to address the issues and incorporate the design insights from the user study.

To address the issue of having concrete feedback on the authoring activity, the authoring interface was connected with a concrete virtual world, Second Life so that users can view the created behaviors and find any problems with them. In order to simplify the steps required to link behavior condition to lower level percepts,the notion of complex triggers that can be defined using mathematical operators over the basic sensory information was removed. A base set of triggers that could be used individually or combined together to make more complex triggers was provided to the users. In order to simplify behavior conditions, the users were asked to create only one condition (i.e. activation condition). In order to present the authoring task as a sequential one instead of deep hierarchies the authoring task was presented as

creating a left to right sequential sequence of actions represented on a timeline.

I have also discussed the authoring task that now involves the following steps. First, user creates a new scene followed by adding the player he would like to have as part of the scene. Next, the user starts creating behaviors by adding triggers for the behavior followed by adding the basic actions that the avatar will perform on the timeline. Once the behaviors are created, user can go in Second Life and view the behaviors by clicking on the attachment on the avatar. Users use a buttoned interface to interact with the avatar in Second Life. Once the interaction finishes, user is presented with a list of questionnaire asking the user to rate the performance of the avatar's behaviors and scenes.

I have also presented the system architecture that consists of four core components a) Second Mind database which is a MS-SQL server based database that stores scenes, behavior elements such as triggers, basic action and interactive responses and b) Behavior Execution Engine that loads the behavior elements and executes them in the external world and c) Web Interfaces that provides the interfaces for creating user account, scenes and behaviors as described earlier and d) Behavior Recommendation Engine provides suggestions during authoring on behaviors that can be used as a starting point during the behavior creation process and one application-specific components which is a) interface layer which provides the ability to connect Second Mind to the simulation platform, such as Second Life or other virtual world. The connection to Second life is created through a collection of scripts that can be downloaded from Second Life marketplace.

Now that I have covered the authoring interface, in the next chapter I discuss the behavior repair approach.

# CHAPTER V

# AI BEHAVIOR REPAIR APPROACH

Once the behaviors are constructed, they would have bugs in them. The problems in behaviors cause a break in player experience when they repeatedly fail. In order to support the novice users in identifying problems with the authored behaviors, a repair approach is provided that identifies issues with the created behavior sets. In this chapter, we detail on the repair approach and provide answers to the research questions 2 and 3 presented in chapter 1.

## *5.1 Overview*

One of the key problems in repairing the behavior sets is to define a metric to measure the success of the authored behavior sets and overall interaction. The work presented in this thesis approaches the problem by defining constraints on the behaviors and the overall interactions that provides a way to measure the success of the behavior sets and the overall interaction respectively. Players provide feedback at the end of their interaction by answering questions related to behavioral performance in relation to these behavioral constraints. Players also provide feedback on their overall interaction. These numerical feedbacks help measure the success of the behaviors and the overall interaction in relation to the associated behavioral and overall interaction constraints respectively. The repair is then carried out through comparison of successful and unsuccessful traces. The comparison provides a way to identify failures in the behavior sets. Once the failures are identified, their modification involves using execution traces (where these failures do not exist) to suggest modifications to the author. In the next sections, I detail on this approach for repairing the authored behavior sets.

## *5.2 Research Issues*

In Chapter 1, I presented several dimensions to the problem of repairing AI behaviors which makes it an especially hard and interesting problem. Lets revisit them within the context of the repair approach. These dimensions are the following:

- **Performance Metrics:** there are several dimensions along which AI performance can be improved and the repair approach could try to improve things along these. Examples of these dimensions include a) believable character performance, b) strong AI Performance, c) adaptation to challenge level and d) improved player experience. In this work, the performance metric that has been used is improvement in player experience. The adaptive approach identifies problems with the behavior sets and adapts them to improve avatar's performance in terms of improving player experience who interacts with the avatar.

- **Performance Measurement Stage:** there are several stages at which the need for repair may be detected, either during the performance of the AI system, in the game episode itself or after the completion of the game episode. In this work, the need of adaptation is detected after the completion of the interaction episode. Once the player has finished the interaction with the avatar, he provides feedback that provides a way to measure if the repair is needed or not.

- **Feedback Agent:** there are several agents (game environment, game designer or player) who could provide the feedback on whether the AI performance is according to the desired performance metric. In our current work, the player is used as the feedback agent.

- **Type of Feedback:** there are several types of feedback that can be provided to the agent. It could be in the form of a boolean value indicating success

or failure, another preferred solution or a complete trace of how the preferred solution was produced. In our current work, the feedback that is provided by the player is in the form of numerical ratings on overall performance of the avatar as well as on individual behaviors.

- **Modification of AI:** there are several stages during which the repair could be carried out, during the performance of the AI system itself or after the completion of problem solving or some combination. In this work, repair is carried at the completion of the player interaction with the avatar.

- **Responsibility of Modification:** the responsibility of repair could be divided between the AI system and the system designer, fully autonomous or shared between the system and the designer. In this work, the responsibility of repair is with the AI system as well as shared between the AI system and the designer. Users can look at the suggested repairs from the AI and can change/improve on them.

The repair approach also needs to address three issues in order to solve the problem of repairing the behavior sets. The first issue is:

**Detecting the need for repair of AI behaviors** In order to identify whether the behavior sets need to be repair, the repair approach needs to recognize when the behaviors have problems. It requires some type of knowledge that can enable it to evaluate the success and the failure of the behavior sets; a specification of what constitutes a successful player interaction, both in terms of what its final results should be, as well as, what its intermediate steps is needed. In our approach, constraints associated with the behaviors and the overall interaction provide a way to measure the success of the behavior sets and overall interaction respectively. As discussed earlier, players provide feedback at the end of their interaction by answering questions generated based on the associated constraints with the behaviors and the overall

interaction. These numerical feedbacks help measure the success of the behaviors and the overall interaction in relation to the associated behavioral constraints and overall interaction constraints respectively. The players also provide feedback on the overall interaction.

Once the need for repair is detected, the repair approach needs to identify the failures which are causing the problem. Therefore, the second issue that the repair approach needs to address is the following:

**Failure Identification:** The repair system also needs to localize the fault points in order to find behaviors that need to be repair. In our previous work towards failure detection, we have addressed this problem through a set of pre-compiled patterns of failures that were used to identify the failures by identifying instances of these patterns in the execution trace [115, 73]. These failure patterns simplified the blame-assignment process into a search for instances of the particular problematic patterns. This method of detecting failures had the advantage that it avoided the potentially very expensive search of looking for failures in large traces. This method however had certain limitations. First, the failure patterns had to be either created by the designers of the behavior repair system or by the users through an IDE[110]. In the first case, the designers had to think of all possible failures that could happen in the game. In a real game, however, it is not feasible to anticipate all possible failures. As a result, the success of the behavior repair system is dependent upon thinking of all possible ways in which the behaviors could fail. In the second case, the burden of creating failure patterns lies on the users of the behavior repair system. In case of novice users in second Mind, this process is cumbersome. In this work, this issue is addressed by repairing the behaviors through identification of differences between successful and unsuccessful execution traces. The comparison provides a way to identify failures in the unsuccessful behavior sets and avoids the authors having to create failure patterns themselves.

Once a set of failures is identified, they need to be appropriately revised. The third issue is therefore the following:

**Behavior Repair:** Once the cause of the failure is identified, the AI agent needs to repair the behavior sets. Having identified some potential cause for its failure, the system needs to find the repair method(s) appropriate for remedying the problem at hand. In our current work, once the failures are identified, the repair involves looking at execution traces (where these failures do not exist) to suggest modifications to the author. The modification could be in the form of for example, adding a new behavior to the behavior sets, or modifying a trigger associated with the existing behaviors.

Next, I present the repair approach in detail.

## *5.3 Repair Approach*

The repair approach consists of the following three parts:

- **Behavioral and Scene Constraints:** authored behavior sets and scenes have associated constraints that provides a way to measure the success of the behavior sets and overall interaction.

- **Player Feedback:** players provide feedback at the end of their interaction. The players provide feedback by answering questions related to behavioral performance in relation to the behavioral constraints.

- **Failure Detection and Repair Approach:** comparison of successful and unsuccessful execution traces allows identification of failures in the unsuccessful behavior sets [73]. The repair then involves looking at execution traces (where these failures do not exist) to suggest modifications on the failed behaviors to the authors in the authoring interface.

Next, Lets look at each of them in more detail.

### 5.3.1 Behavioral and Scene Constraints

Behavioral and Scene constraints provide a way to associate success conditions with the behaviors and overall interaction. These constraints are thus similar in spirit to the post conditions used in PDDL notation[1] notation and success conditions used as part of ABL (a behavior language) [67]. Post conditions in PDDL are used to represent a set of conditions that are established after the behavior is performed. ABL (a behavior language) developed by Michael Mateas used success conditions as a way to check from the world state whether the behavior was able to achieve the desired purpose or not. These are conditions in the world that would become true as a result of success of the behavior and can be tested for from the world state(s) obtained after the execution of the behavior. In the case of behaviors and scenes defined for personality rich characters or characters that are performing a particular role (like an island shopkeeper or a museum host in Second Life), success conditions incorporate personality and/or role specific metrics. Behavior constraints serve that purpose. Behavior constraints provide a measure of how the character should be performing his behaviors in accordance with his personality or role. These constraints are used a way to check through some feedback (like player's feedback, for example)) whether the behaviors came out the way as it was envisioned or not when the constraints were assigned to it by the behavior author.

The constraints associated with the behaviors in the current authoring interface are the following:

- Enthusiastic
- Happy
- Positive

[1]Planning domain definition language (PDDL) was developed by Drew McDermott in 1998

- Pessimistic
- Pushy

The constraints associated with the overall interaction in the current authoring interface are the following:

- likeable
- good salesman

All the scenes are associated with two constraints by default, a) overall experience and b) avatar's performance. As described earlier, during the authoring process, the author uses a set of sliders to specify values (on a 5 point scale) for the associated constraints with the behaviors and the scenes. For example, the author could specify that he wants the shopkeeper to be highly enthusiastic (setting a high value on the slider) and less pushy (setting a low value on the slider) when performing the greet behavior.

### 5.3.2 Player Feedback

The player feedback is used as a way to measure the performance of the behavior set with respect to the author specified behavioral constraints and overall interaction constraints. As described earlier, the player feedback is collected by answering Likert scale questionnaire. The feedback is stored along with the execution trace of the player interaction. As different authors use the authoring system and multiple players interact with the authored avatar(s), various execution traces (and the corresponding player feedback) corresponding to their interaction is collected. The performance of the authored behavior sets with respect to the associated behavioral constraints and overall interaction constraints varies; some behavior sets would be better able to satisfy the overall constraints(called successful traces) and some others would not be able to satisfy the overall constraints (called unsuccessful traces). As multiple players

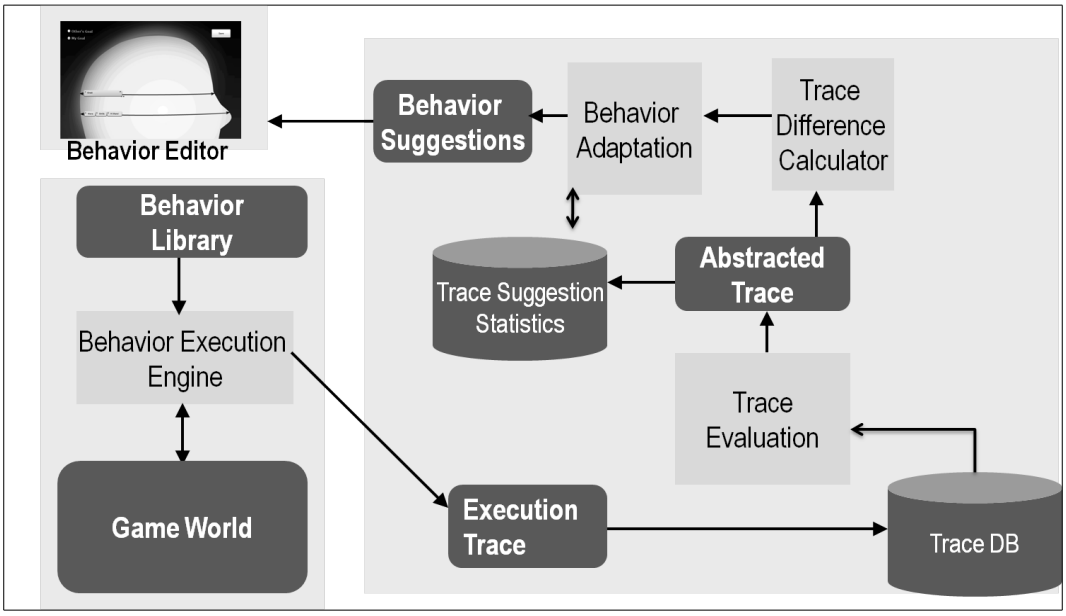



**Figure 30:** The figure shows the behavior repair architecture

interact with avatars, various successful and unsuccessful traces are collected. These traces provide a way to detect failures that cause the unsuccessful behaviors and the overall interaction to not achieve the associated behavioral and overall interaction constraints. They further help in suggesting the necessary modification(s) to repair the failed condition. Let us discuss this failure detection and repair mechanism next.

### 5.3.3 Failure Detection and Repair Approach

#### *5.3.3.1 Trace Evaluation*

The behavior execution module records an *execution trace* as the player is interacting with the avatar. The execution trace is used to record important events happening during the interaction with the avatar. These events contain information regarding the behaviors that executed, their start times and the times at which user clicks the buttons, for example (shown in Figure 31). The trace also contains the external state of the world recorded at various intervals so that different information can be extracted from it during reasoning about the failures happening at execution time. The game state is also shown in Figure 31. The trace provides considerable advantage

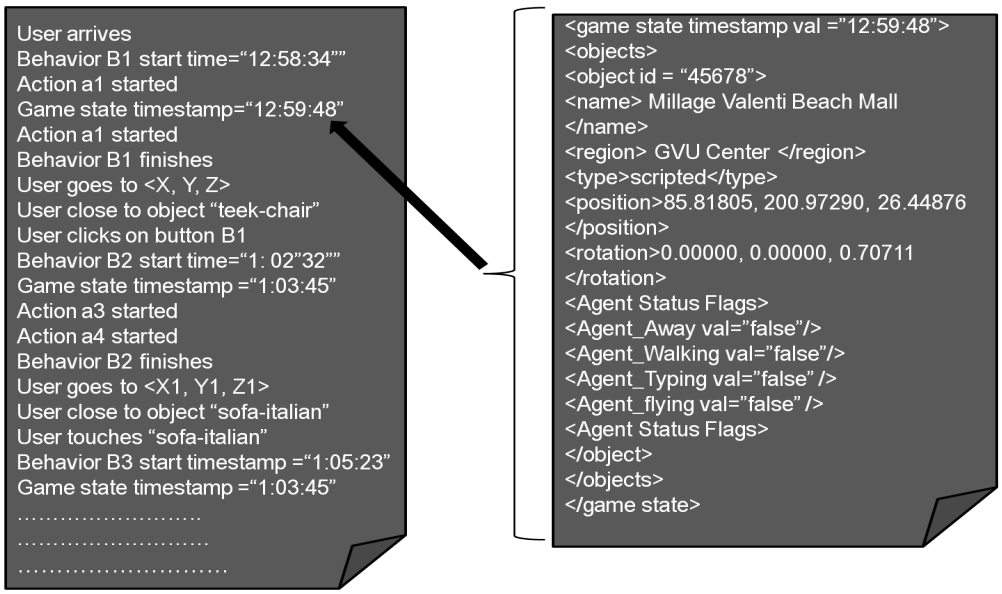



**Figure 31:** The figure shows an example execution trace (left) and example game state (right)

in performing repair, since the trace can help localize portions that could possibly have been responsible for the failure. As multiple players interact with the avatar(s), more execution traces anplayer feedbacks are recorded and stored in the *TraceDB* database.

The *Trace Evaluation* component uses the execution traces stored in the database to abstract things from it. An example abstracted trace is shown in Figure 32. The abstracted traces are used by the rest of components of the behavior repair. Some of the information contained in the individual abstracted traces is the following:

- **Executed behaviors:** information regarding behaviors that successfully executed during the interaction.
- **Unexecuted behaviors:** information regarding behaviors that didn't get executed during the interaction.
- **Time interval between user turns:** the time between two consecutive user clicks. Please note that user input comes in the form of clicking on buttons.

- **Avatar initiated behaviors:** among executed and unexecuted behaviors, behaviors that are triggered by avatar himself.
- **User turns:** the buttons that the user clicked during his interaction as well as the buttons he didn't click on.
- **Behavior execution order:** contains information on the order in which the behaviors were executed.
- **Scene and behavior rating:** contains information on ratings for the scenes and the behaviors that were executed.
- **Gesture count:** contains information regarding number of gestures performed during the interaction and for each executed and unexecuted behavior as well.
- **Verbal actions count:** contains information regarding number of verbal actions performed by the avatar during the interaction and for each executed and unexecuted behavior as well.
- **User events:** contains information regarding time of arrival, departure time of user and user's movement events during the interaction.

At this stage, all the information contained in the abstracted traces for a particular scene is combined together as well (shown in Figure 33, in a structure called Trace Combination Abstraction (TCA). Some of the metrics that are included in the combined information in TCA are the following:

- **Average executed behaviors:** keeps track of the behaviors and average number of times they successfully executed across all the player interactions for a scene of an avatar.

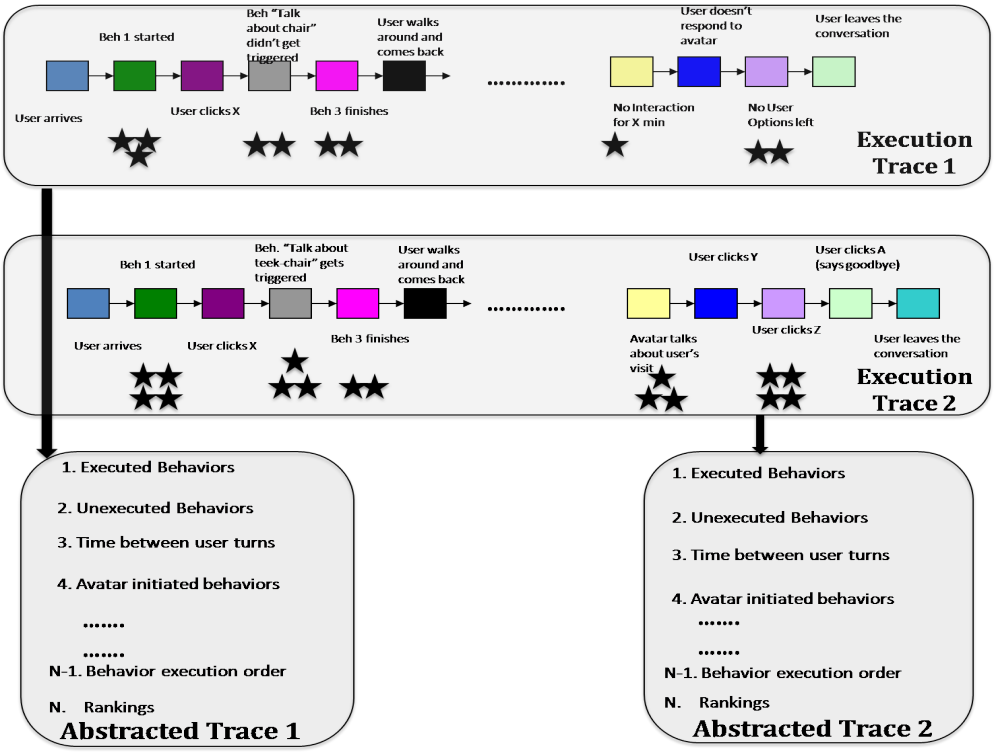



**Figure 32:** The figure shows the process of abstracting information from the execution trace

- **Average unexecuted behaviors:** keeps track of the behaviors and average number of times they didn't get executed across all the interactions for a scene of an avatar.

- **Average time interval between user turns:** keeps track of average time interval between user turns across all the interactions for a scene of an avatar.

- **Average avatar initiated behaviors:** keeps track of all the avatar initiated behaviors and their average number among executed and unexecuted behaviors.

- **Average user turns:** keeps track of all the buttons that the user clicked as well as the buttons he didn't click as well as their average count across all the interactions for a scene of an avatar.

- **Average behavior execution order:** keeps track of all the behavior execution

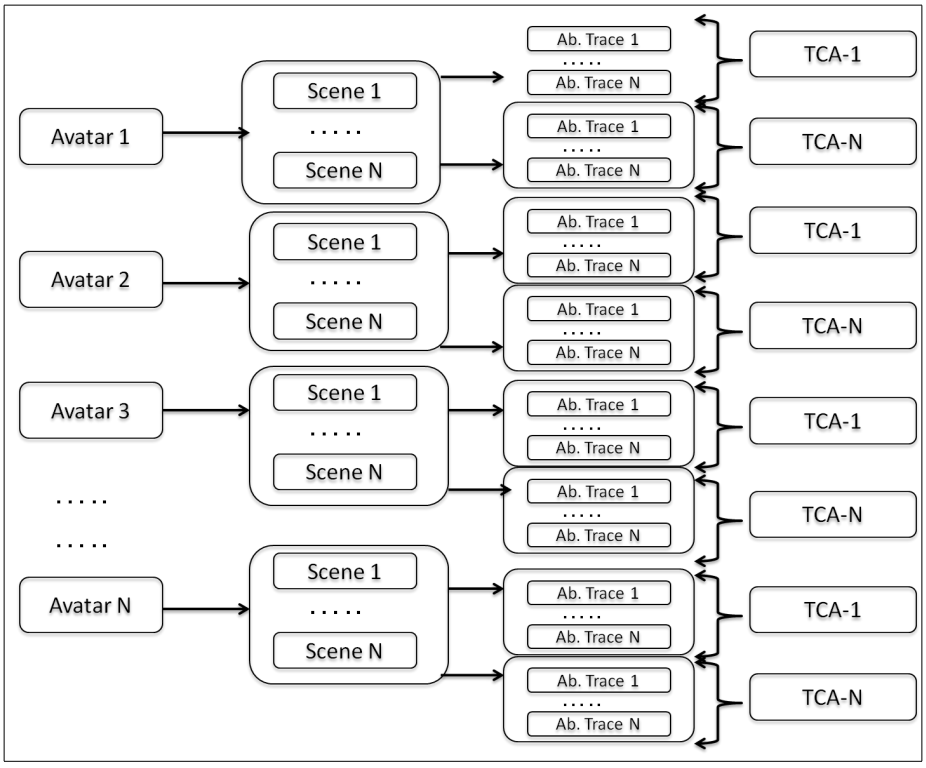



**Figure 33:** The figure shows the process of abstracting information from the execution trace

order and their average count across all the interactions for a scene for an avatar.

- **Average scene and behavior rating:** keeps track of average scene and behavior ratings across all the interactions for a scene of an avatar.
- **Average Gesture count:** keeps track of the gestures and their average count across all the interactions for a scene of an avatar.
- **Average verbal actions count:** keeps track of the verbal actions and their average across all the interactions for a scene of an avatar.

Once the *Trace Evaluation* component has performed the abstraction and generation of TCA, this information is sent to the *Trace difference calculator* component. This information is also stored in the database *Trace Suggestion Statistics* so that it need not be calculated again when more data comes in the form of newer execution traces and player feedbacks and the behavior repair is performed again.

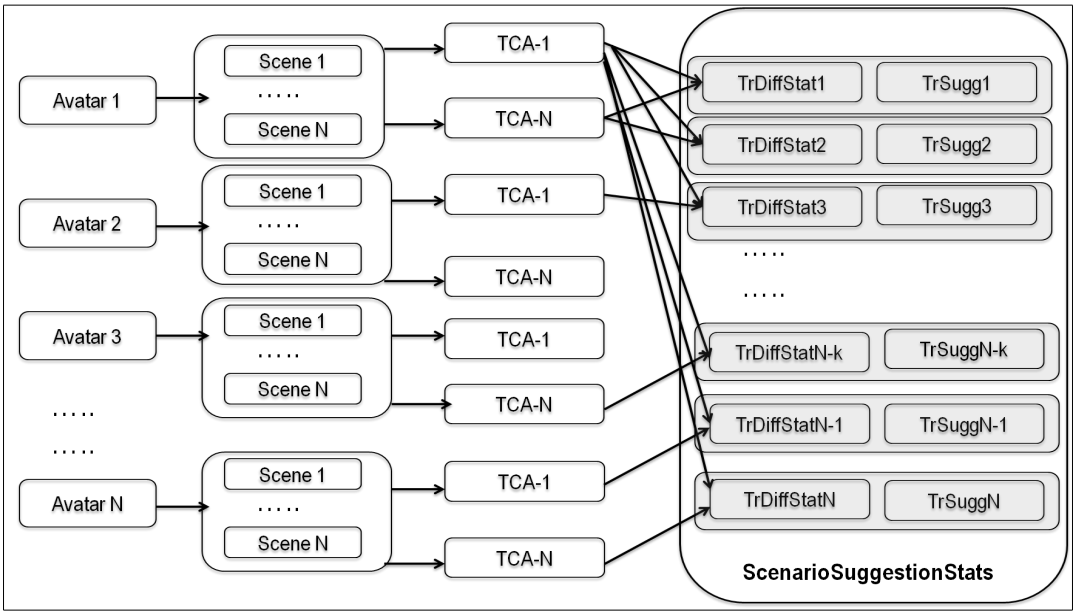



**Figure 34:** The figure shows the process of generating differences and suggestions by the behavior repair module

#### *5.3.3.2 Trace Difference Calculator*

The *Trace difference calculator* records the differences based on comparison of a) abstracted traces for a particular scene of an avatar among themselves, b) combined abstracted traces, TCA that belong to different scenes of an avatar among themselves and c) combined abstracted traces TCA of scenes belonging to different avatars among themselves. This comparison process is reflected in Figure 34. In order to do this difference identification, abstracted traces and the combined abstracted traces (TCA's) are also marked as successful or unsuccessful at this stage. In order to mark combined abstract traces TCA as successful or unsuccessful, a simple ranking scheme that is based on the overall interaction ranking (corresponding to the default overall interaction constraints) provided by the player at the end of the interaction is used. The interaction rankings across all the traces for a particular scene for an avatar is averaged out to determine and mark TCA as successful or unsuccessful. For abstracted traces, the ranking provided by the player on the default constraints associated with the overall interaction is used. Trace differences are performed across pairs that have

a ranking difference of at least 0.5 or higher. The higher ranked trace within these trace pairs is termed as successful and lower ranked termed as unsuccessful. Some of the differences that are detected by the *Trace difference calculator* are the following:

- **Actions level difference:** holds the difference in the executed actions across the two traces. These differences are for example, differences in expressivity of behaviors like more gestures present across traces or differences in verbal actions. This difference captures things like a more expressive avatar resulting in variation in ratings from the player.

- **Non-Existent matched behavior difference:** holds better matched behaviors (with respect to the associated behavior constraints) that are present in one trace and absent in the other. This difference is calculated when comparing traces that belong to similar scenes for different avatars. This difference captures interesting interactions (in the form of behaviors) that could be present in one trace and not present in another.

- **More user option difference:** holds differences in user options across traces. This difference is calculated when comparing traces that belong to similar scenes for different avatars. This difference captures interesting user options (and thereby user interactions) that could be present in one trace and absent in another.

- **Better constraints match difference:** holds difference in terms of similar behaviors that would have a better constraint match in one trace compared to the other. This difference is calculated when comparing traces that belong to similar scenes for different avatars. This difference captures behaviors that have better matching constraints (with respect to player ratings) compared to other similar behaviors where the constraints might not match.

- **User click count difference:** holds differences in terms of number of user clicks during interaction. This difference is calculated when comparing traces belonging to similar scenes for different avatars as well for traces belonging to same scene for an avatar. This difference captures differences in user interaction (captured in the form of user button clicks).

- **Unexecuted behaviors difference:** holds differences in similar behaviors that didn't get executed in one trace but got executed in another. This difference captures problems in behaviors in the form of trigger conditions which could have resulted in untriggered behaviors.

- **User turns time interval difference:** holds differences in terms of significant time intervals between user turns across the two traces. This difference captures problems caused due to no interaction with the user for significant periods of time.

- **Executed behaviors difference:** holds differences in terms of similar behaviors that got executed in the traces.

- **Avatar initiation difference:** holds differences in avatar initiated behaviors across traces. This difference captures if a more proactive avatar could be a factor in causing a difference in player rating.

#### *5.3.3.3 Behavior Adaptation*

Once the differences are identified across TCA and abstracted trace pairs, they need to be addressed through appropriate modifications. Our previous work in this area involved having a collection of modification routines associated with the failures that provided the necessary set of modifications to repair the failed conditions. This suffered from the limitation that the appropriate modifications had to be defined beforehand [115, 73]. The current work involves looking at the traces (where these

**Table 3:** The table shows some of the differences and the repair methods used to eliminate those differences

| **Differences** | Repair Method |
|---|---|
| **Actions level difference** | Pick actions from other similar behaviors from successful traces |
| **Non-Existent matched Behavior difference** | Pick non-existent behaviors from the other trace (successful or unsuccessful) |
| **More User Option difference** | Pick user options from other successful traces |
| **Better constraints match difference** | Suggest better constraint matching behaviors from other trace (successful or unsuccessful). |
| **User click count difference** | Suggest avatar initiative taking behavior with appropriate triggering condition based on time interval. |
| **User turns time interval difference** | Suggest avatar initiative taking behavior from successful traces.<br>Create avatar initiated behaviors based on the unsuccessful trace and suggest them with appropriate triggering condition based on behavior_finishes trigger. |
| **Unexecuted behaviors difference** | Use triggers from similar executed behaviors in other trace (successful or unsuccessful).<br>Loosen triggers associated with the unexecuted behavior |
| **Avatar initiative difference** | Pick high ranking avatar initiative behaviors from other traces (successful or unsuccessful).<br>Create avatar initiative behaviors based on the current trace and suggest them |

failures do not exist) to identify the right way to modify and eliminate the differences. Table 3 shows some of the methods to modify and repair the differences.

As *Trace difference calculator* calculates differences for successful and unsuccessful TCA and abstracted trace pairs respectively, in the next step, *behavior adaptation* then identifies appropriate suggestions using the repair methods corresponding to a difference shown in Table 3. These suggestions are recorded along with the rankings of these suggestions. The ranking used for the suggestion is the same as the rating of TCA or abstracted trace (calculated as part of marking them as successful or unsuccessful). In some suggestions, this ranking is combined with the behavior constraint match score which is given more weightage (three times). This is done

in order to identify and eventually use suggestions that might be present in certain TCA's and abstracted traces that do not have high overall interaction rating but still might have good suggestions. For example, suggestion corresponding to "Better constraints match difference" and "Non-Existent matched behavior difference" would have a ranking that is not only based on the overall interaction ranking but also based on the match of the behavior's constraint with the player's rating. The suggestions are recorded along with the differences (as shown in Figure 34) in a structure (called *ScenarioSuggestionStats*) corresponding to a scene for an avatar.

As *Trace difference calculator* calculates more differences for abstracted trace and TCA for a particular scene of an avatar based on comparison with other abstracted traces and TCA, the difference list grows. As these differences are calculated, their presence or absence across successful and unsuccessful trace comparison is also recorded as well. As these differences are identified, the *behavior adaptation* generates more suggestions for the identified differences based on looking at new traces. The suggestion list for a particular difference would eventually have multiple suggestions in it. For a particular difference, the ranking of the successful and unsuccessful abstracted trace or TCA whose comparison resulted in the difference is also recorded as well. The differences which are more commonly identified to be present based on comparison of successful and unsuccessful traces are potentially selected as the ones that need to be addressed. The difference in rankings for the two traces (successful and unsuccessful) acts as another filter to identify the differences to address. The difference in rankings for the two traces is used in order to give more weightage to the differences that are identified based on comparison of higher ranked successful and lower ranked unsuccessful traces.

To repair a particular selected difference, a ranking scheme is used to pick a suggestion from the suggestion list. The ranking is based on the usage of the suggestion i.e. how many times it has been selected to remove the differences and the suggestion

ranking. The highest ranking suggestion is then selected for repairing the difference and eventually presented to the author in the authoring interface.

**Illustrated Example:** Lets look at an example to understand the inner workings better. Lets take the example of three different avatars (with three different behavior sets) playing the role of island shopkeeper selling furniture on a SL island. Once the behaviors are authored, multiple players interact with the avatars and provide feedback on the behavior and overall interaction constraints at the end. As multiple players interact with the avatars, multiple execution traces and player feedbacks are collected for the scenes corresponding to the three avatars.

The *Trace evaluation* first abstracts out the execution trace and then also generates TCA's for the scenes of the three avatars. The *Trace difference calculator* ranks TCA1 as the highest (corresponding to scene 1 or avatar 1), TCA3 (corresponding to scene 1 of avatar 3) ranked second and TCA2 (corresponding to scene 1 of avatar 2) as the lowest ranked. It also finds the following differences based on comparison of successful and unsuccessful trace comparison which need to be addressed:

- **Unexecuted behavior difference:** A particular behavior didn't gets triggered for TCA1. A similar behavior type gets triggered in TCA2 and TCA3.
- **Avatar Initiative difference:** In TCA1 and TCA3, the avatar takes more initiative to talk about certain things.
- **Non-existent matched behavior difference:** In TCA1 two behaviors were better matched and were not present in TCA2.

As these differences are identified, traces (where these failures are not present) are used to identify the modifications needed to correct the failed conditions. The first difference refers to the fact that a particular behavior type got triggered in TCA2 and TCA3 but didn't get triggered in TCA1. The suggestion for this failure is to look at TCA2 and TCA3 and pick the triggers and use them as suggestions

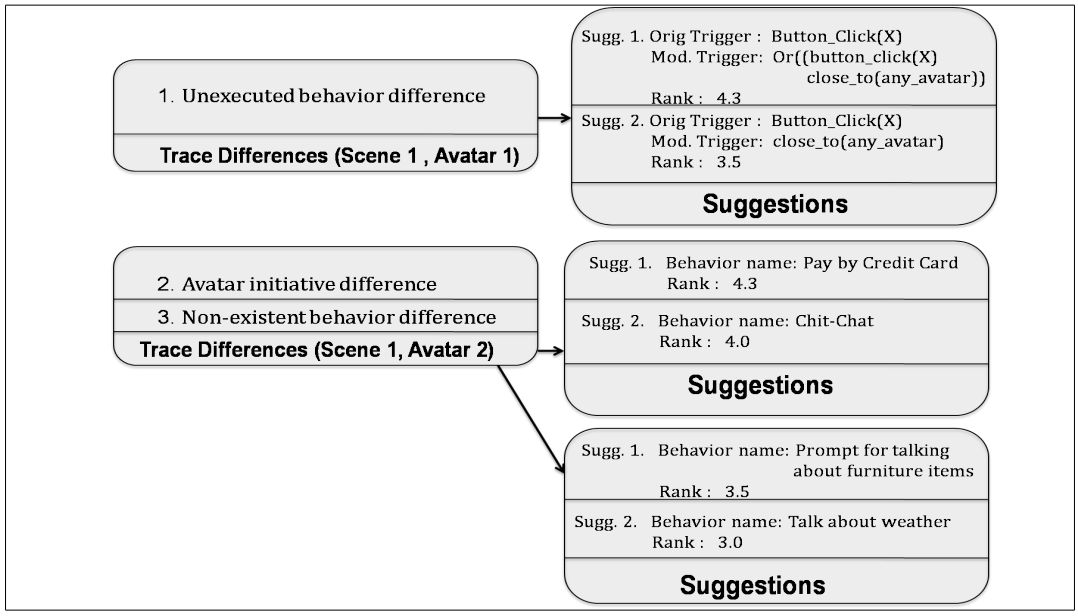



**Figure 35:** The figure shows the behavior suggestions produced for the example discussed in text.

for untriggered behavior in TCA1. For the second difference where the avatar has taken more initiative in TCA1 and TCA3, the *behavior adaptation* would add more initiative taking behaviors for TCA2 using examples from TCA1 and TCA3. For the third difference, two behaviors, payment by credit card and chit-chat behavior are present in TCA1 and are suggested. Figure 35 shows the suggestions corresponding to the differences.

#### *5.3.3.4 Behavior Suggestions*

There are three categories of suggestions that are generated by the behavior adaptation module that are then presented to the author. The three categories are the following:

- **Trigger suggestions:** provides suggestions on how to modify existing triggers for the behaviors. Figure 36 shows an example of a trigger suggestion. As example of a trigger suggestion is modification of original triggers for the greet behavior. The original trigger is a button_click trigger where the greet behavior

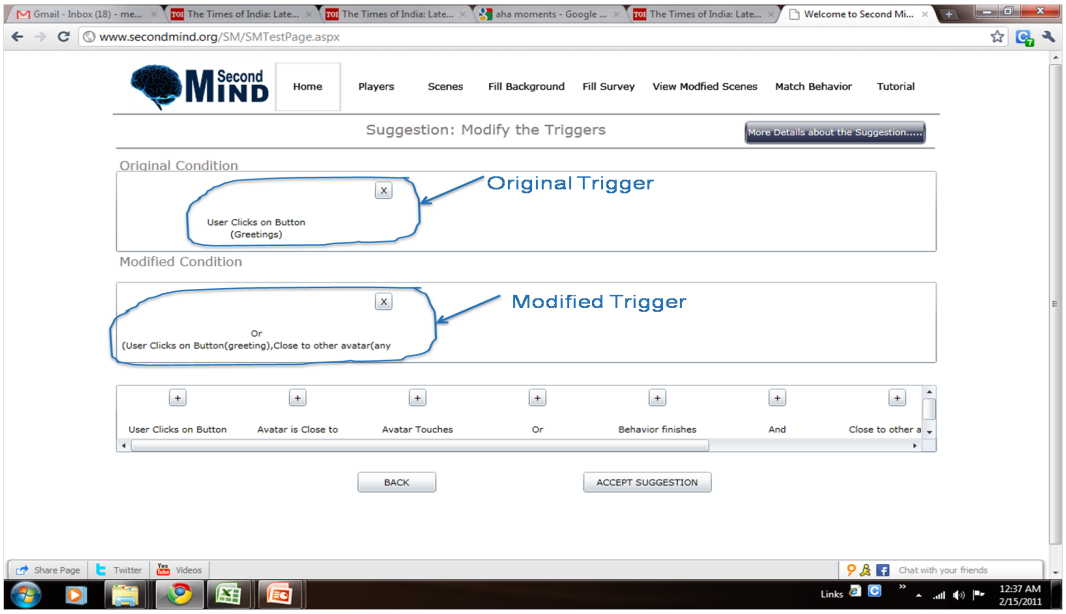



**Figure 36:** The figure shows an example trigger suggestion presented in the authoring interface

would happen when the player greets the avatar by clicking on a button. The modified trigger would now be a combination of button_click trigger with avatar is close to (any-avatar) trigger which would now allow the avatar to perform the greet behavior when he is close to another avatar as well. Authors were also presented with reasoning on why a trigger suggestion was made. The reasoning that was presented to them was the following:

> *In interactions with the user, it was observed that the behavior is sometimes not performed/triggered by the avatar due to the current triggers that have been defined. Other similar behaviors that have received higher ratings and have performed seemed to have the modified trigger that is suggested.*

- **Existing behavior suggestions:** provides suggestions on how to modify existing behaviors by suggesting replacement behaviors. Figure 37 shows an example of such a suggestion. An example would be a replacement behavior for a greet behavior which originally didn't have a wave gesture. The replacement behavior

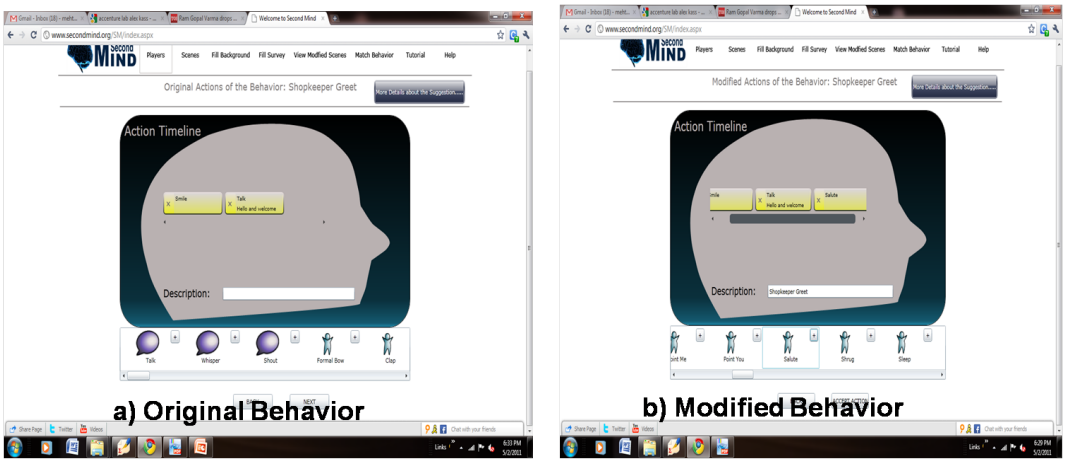



**Figure 37:** The figure shows an example action suggestion presented in the authoring interface

will now have a wave gesture added. Authors were also presented with reasoning on why a suggestion was picked. The reasoning that was presented to them was the following:

> *Other similar behaviors that have received high ratings with the users, it was observed that behaviors had different actions than what are there in the behavior defined by you. It is highly recommended you add these new actions.*

- **Behavior addition suggestions:** provides suggestions on new behaviors that could be added to the scene. Figure 38 shows an example of a new behavior suggestion. An example of an addition suggestion is a payment method for a shopkeeper scene which allows the player to pay for his purchases. Authors were also presented with reasoning on why an addition suggestion was made. The reasoning that was presented to them was the following:

  > *In other similar scenes created by other users, it has been found that Behavior XX [name of the suggested behavior] has been rated highly. Overall scene ratings are also improved with this behavior.*


**Figure 38:** The figure shows an example new behavior suggestion presented in the authoring interface

### 5.3.4 Chapter Summary

In this chapter, I have looked at the repair approach to identify and repair problems with behaviors. I have presented the approach that helps identify problems with the authored behaviors. The approach consists of three parts a)Behavioral Constraints that are associated with behaviors that provides a way to measure the success of the behavior sets, b)Player Feedback where players provide feedback at the end of their interaction. The players provide feedback by answering questions related to behavioral performance and overall interaction in relation to the behavioral constraints and overall interaction constraints respectively and c)Failure Detection and Repair Approach where comparison of successful and unsuccessful execution traces allows identification of failures in the unsuccessful behavior sets [73]. The repair then involves looking at execution traces (where these failures do not exist) to suggest modifications on the failed behaviors to the author.

A four part evaluation study was conducted on the authoring and repair approach that I describe next.

# CHAPTER VI

# EVALUATION STUDY

## *6.1 Overview*

In the previous chapters, I have introduced the behavior authoring and the repair system. In this chapter, I discuss the approach used for evaluating them. I also discuss the results from the evaluation study. The evaluation study explicitly avoids the approach of testing one system directly against another. Instead, the approach that has been used by us is to evaluate Second Mind against a certain standard of performance than against some other software artifact. The goal was that Second Mind can be considered successful if it is understood and used by a significant number of non-programmers, a level of performance that is considered to be a significant improvement over the current state of affairs.

In order to evaluate the approach, a four phase evaluation study was conducted. In the first phase of the evaluation study, six category of users (we use the term authors for them) were invited to create shopkeeper behaviors for an avatar on a Second Life island using the authoring system. The six category of authors were the following:

- User Interface(UI) designers
- AI behavior programmers (i.e. authors who had created AI behaviors in games in the past)
- Non-Programmers (from different backgrounds without any design, programming or AI behavior programming experience).

- Second Life non-programmers (who were used to the Second Life interface but still had no programming background)
- Acting background non-programmers
- Programmers (with experience in Java)

The first phase was guided by part of the research question 1 i.e. whether the design solution presented in Chapter 4 allows novice users to easily construct AI behaviors. In the second phase, another set of users (we use the term players for them) interacted with the shopkeeper avatars (created as part of the first phase) in SL. This provided the necessary feedback for AI repair system to improve the behavior sets as well as gather the necessary ratings on performance of the authored behavior sets. In the third phase, three category of authors (Second Life non-programmers, Acting background non-programmers, and AI behavior programmers) who carried out behavior authoring in the first step were invited back. During this phase, first the authors were presented with text based feedback on how well the behavior sets performed. The text based feedback contained simple textual description detailing the performance of the behaviors with respect to associated constraints. The text based feedback was based on feedback provided in second phase. Secondly, the authors were presented with the suggested changes from the AI repair system. Authors used these suggestions as starting point and were free to modify the behaviors based on the suggestions. In the fourth phase, another set of players interacted with four sets of behaviors. The first set was the behaviors that were originally created by the authors in phase I. The second set of behaviors were the ones adapted by the authors using the numerical rating shown to them in Phase III. The third set of behaviors were the ones that have been modified based on AI repair suggestions (without any author intervention). The fourth set of behaviors were the ones that have been modified in collaboration between AI and the authors. Players interacted with these four

behavior sets in Second Life and provided feedback on their interaction. The ratings on these behavior sets provided a way of measuring and comparing AI repair approach effectiveness in appropriately repairing the behavior sets and achieving a better player experience. The third and fourth phase were guided by part of research question 3 i.e. whether the system repairs failures in human authored AI behaviors to improve player experience.

Let's look at each of the phases and the results obtained in more detail:

## *6.2 Phase I study*

**Research Questions:** In the first phase of the study, different categories of authors were invited to understand their experience in terms of authoring system limitations and things that worked for them while creating AI behaviors. This study was guided by the broader research question 1 introduced in Chapter 1 i.e. whether the design solution allows novice users to easily create AI behaviors. Within the scope of this broader question, we had the following specific questions:.

- Can the user conceptualize the authoring process using the user interaction approach employed? What parts of the authoring process are difficult to conceptualize?
- Are there behaviors (and what kind) that the user would have difficulty in creating using the current authoring interface?
- What are the points at which the authoring process becomes tedious?

**Study Setup:** Sixty five participants (referred as P1:P65) were recruited, 20 females and 45 males through craigslist.org. Participants were screened so that they would fall into each of the categories that were described earlier. The participants ranged across multiple races, education levels, and ages (from 22 to 50 with an average age of 28). Players signed a consent form and filled out a background questionnaire

(shown in Section B.6 of the Appendix) at the beginning of the session. The authoring tasks was described to participants before the start of the authoring session. For example, they were told that their goal is to create behaviors for an avatar that would act out as a shopkeeper in a virtual world. They were asked to list the behaviors on a piece of paper. This step was needed so that they can think of behaviors they would want their avatar to do without any understanding of the authoring system and any of its limitation. Next, they were given a tutorial file that showed an example scene and behaviors for it. After they had an opportunity to go through the tutorial (approx. 15 minutes), they were asked to use the authoring system to create the behaviors that they had sketched out for the interaction on the piece of paper. The users had to carry out Steps a-f mentioned in Section 4.2. As the authors started creating the behaviors they had sketched on the piece of paper using the authoring interface, they were asked to highlight any limitations they come across in creating the behaviors they had in mind. During the authoring process, they were also asked to talk aloud about any limitations that they came across in creating the triggers and action sets. These were noted down and discussed during the interview session. The complete authoring session, on an average, lasted for about two hours.

**Data Collection:** In terms of data collection, the background information from the authors was collected as I mentioned earlier. During the authoring session, the players interaction with the authoring system was observed and notable events such as 'unable to fill a condition', or 'having difficulty creating things' were logged. An open-ended interview was conducted about their authoring experience at the end of the session. The questions that served the basis for the interview are listed in Section B.7 of the Appendix. The authors were also asked to fill out quantitative questionnaire (shown in Section B.5 of the Appendix) at the end of the interaction. During the authoring activity, various quantitative information like time spent in creating and correcting the different behaviors and time spent on various aspects of

behavior creation process like creating the trigger, gestures, verbal actions etc was noted down as well.

**Data Analysis:** To analyze this vast quantity of interview data from all the studies, a combination of qualitative and quantitative analysis was used. The user responses from the interviews and observed user actions during the authoring process were transcribed. The data obtained was analyzed, focusing on a qualitative analysis of the results. In order to perform a qualitative analysis well-known qualitative analysis method, Grounded Theory [104] was employed. Using grounded theory principles, notes were made for user responses from the interview. The transcriptions were processed, first to take open-ended notes and second to highlight more salient ideas expressed by user. Codes, such as 'User is unable to fill precondition' and 'User was not able to complete scene creation process', were generated and multiple user quotes were listed for each idea in an Excel spreadsheet. Within the spreadsheet, I then conducted data analysis where related codes were grouped together. For example, all codes that had something to do with problem of creating behaviors were initially grouped together into related concepts. I went through an iterative process of describing each concept and conceptually linking them to each other. I formed a single hierarchy of the concepts, went back to the spreadsheet and organized the same concept along with the top two to three quotes for each concept. This hierarchical organization of concepts served as an outline for the qualitative findings. A subset of the codes are shown in Table 4. In total, we had about thirty phenomena codes, such as 'Player talks about system supporting creativity' and 'Player feels authoring system was easy to use' listed into an Excel files with multiple player quotes for each phenomena. This hierarchical organization of themes served as an outline for writing. I also conducted one-way analysis of variance (ANOVA) tests and Scheffé post-hoc tests to analyze various quantitative measures across the various author categories (shown in section B.1 in Table 10 and section B.3 in Table 11 in the Appendix).

**Table 4:** The table shows some of the codes used for data analysis of phase I

| **Category** | Player Comments |
|---|---|
| **Finds the interface easy to use** | The scenes part and the choices that were given regarding what a shopkeepers could do was easy. The interface seemed pretty easy to use. |
| **Would like to have more animation** | I would like to see more animations. If I had more quality animations then I would be able to get more variety in the behaviors. |
| **Would like to see visualization in Second Mind** | Visualization of the actions would be helpful, being able to see the animations in-browser would avoid going into Second Life. |
| **Find the interface creative** | It seemed like you could do whatever you want. They were so many different ways in which actions could be utilized and combined in many different way. |
| **Likes the timeline editor** | Being able to build things in the order in which you put them in, its very much you click it and its in, this is very convenient |
| **Like he could be subtle with the choices** | I liked you could do it as a talk or shout which means you can get subtle with it. |
| **Feels the possibilities were limitless** | I could see how you can create a new world and spend hours creating a whole world with your imagination. |
| **Likes that he could detail it like an artist** | I think the strongest point is being able to detail it, it could probably have more but just in the little game i created, the emotions and the actions provided option of detailing the character. |

The analysis provided the following results:

- **Programming scaffolding for non-programmers:** Non-programmers were able to author behaviors using the scaffolding provided by the authoring system through a) understandable terminology and representation vocabulary, b) easy to use authoring environment and c) virtual world connection that helped them identify problems with the authored behaviors.
  - **Understandable terminologies and representation vocabulary:** Authors were able to understand the terminology and representational vocabulary (like scenes, behaviors triggers, actions etc). Based on the quantitative ratings and qualitative feedback, non-programmers had an equal understanding of the terminology on par with AI behavior programmers and programmers.
  - **Ease to use of the interaction design approach:** Non programmers found that they did not need any programming experience to use the system and provided numerical ratings (on ease of use) which were statistically similar in rating to AI behavior programming experts. Authors felt that using the timeline editor, the system allowed them to be very creative where they could go in the authoring interface and create in a small amount of time any kind of world that they had imagined.
  - **Virtual world connection with the authoring interface helped authors identify problems with behaviors:** The connection to a virtual world (Second Life) was very helpful for authors in identifying problems with the created behaviors. Authors created the behaviors in the authoring interface and then went into SL, interacted with the authored behaviors and identified any issues with the behaviors.
- **Authoring interface limitations:** Authors requested the following features

to be added in next version of the system a) video tutorials, b) advanced programming features and c) Animation preview option and new animations.

- **Video tutorials:** Authors felt that it would have been very helpful to have more video tutorials that they can just view instead of a tutorial file. The video tutorial would have made the process more easier for them.
- **Advanced programming features:** Authors esp. programmers wanted to have other advanced features like random behaviors or behaviors that could be selected based on probabilities. They also wanted to have features where they could go in and view the internals of the behaviors (like looking at the generated code) to modify it.
- **Animation preview and new animations:** Authors found that not having an animation preview button was a limitation of the system. An animation preview button would have allowed authors to watch the avatar's animation right in the authoring interface itself (currently they had to go in second life and view the results of the animation they had selected). Authors also wanted the ability to create new animations that would have allowed them to create newer kind of behaviors.

In the next sections, lets look at these results in more detail.

### 6.2.1 Understandable terminology and AI vocabulary:

The results from the numerical ratings that the authors provided on their understanding of the terminologies that were used in the system (shown in Figure 39) were analyzed. Numerically, the authors provided a high rating on terminology understanding. On an average, authors provided a rating of 4 out of 5 on understanding the terminology. Based on the results of conducting one-way ANOVA tests, the differences in ratings as shown in Figure 39 across different author categories were not

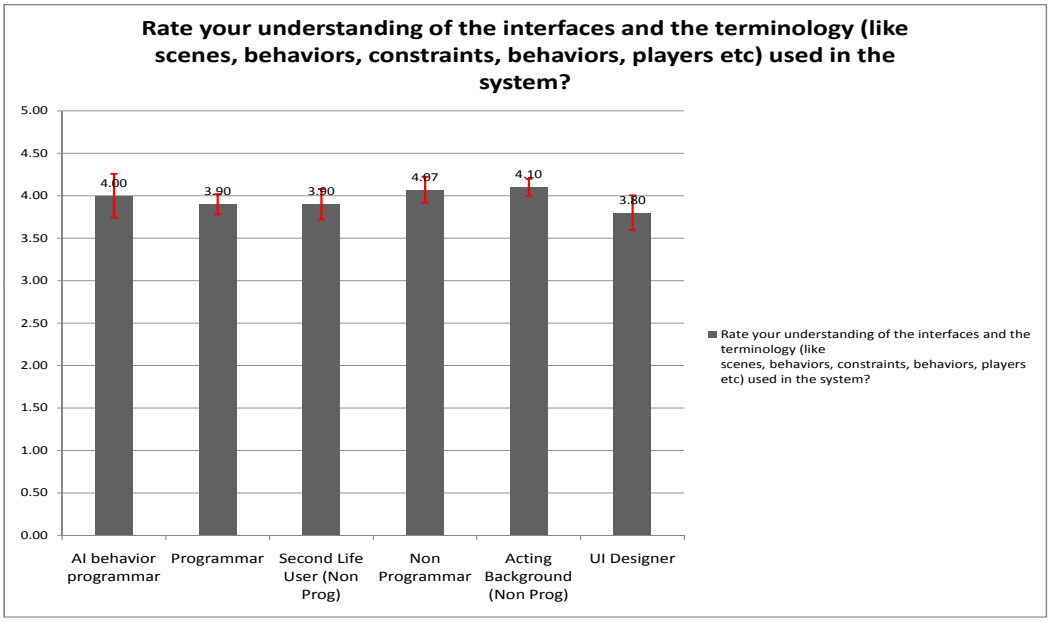



**Figure 39:** The figure shows the author's ratings on terminology understanding. Red bars indicate the standard error.

found to be statistically significant, F(5,59)=1.29, p=0.28. Table 10 in section B.1 shows the mean and standard deviations for various quantitative measures across various author categories. Interview data further qualified the numerical rating. Authors mentioned that they found the terminology to be "very clear and intuitive to understand" [P20]. Authors felt that the terminology was "very easy for non-programmers to understand" [P3] and as a result they were able "to design pretty much everything with little problem" [P14]. P5 expressed his ease of understanding:

*It took a minute but I didn't have any difficulty at all-I was able to understand the terms pretty well and this is coming from someone who has been to work and had quite a long day and I was able to get into it and design pretty much everything with very little problem (P5)*

As part of designing the authoring activity, an underlying representation formalism and terminology was used to represent the authoring process. One of the key things that was planned to be tested through this phase was the understandability of the AI terminology. An AI expert with experience of conducting behavior programming would have the necessary background to understand the terminologies used in the authoring process. However one of the goals was to make sure that the terminology had words, phrases, and concepts familiar to a non-programmer, rather than terms more familiar to an AI behavior programmer or a person with a more technical background. The analyzed results from the interview data and numerical ratings provided by the authors (in Figure 39) indicated that this goal was achieved as the non-programmers had an equal understanding of the terminology terms like (scenes, behaviors, triggers etc) as an AI behavior programmer who was an expert in the domain.

One terminology issue that came up during the interview discussion was the problem in understanding the role of constraints during the initial stages of the authoring process. As the authors started creating the behaviors, initially they had difficulty in understanding the role of constraints. P14 expressed:

> *The constraints were little bit confusing at first. I couldnt initially understand how it affected my character. (P14)*

Once the authors had created a few behaviors, authors went into Second Life and interacted with the authored behaviors in Second Life. At the end of their interaction, authors answered the questions regarding behavior constraints (as discussed in Section 4.2.4), which helped them clearly understand the role of constraints and was in fact found to be helpful. Authors mentioned that assigning constraints to the behaviors as the first step of the behavior creation process made them think deeply about the kind of personality they would want their avatar to have. The constraints made

the authors think about what they would like the avatar to do and the overall user experience as expressed by P4:

*I think the path it leads you through the authoring process, by selecting the traits first, makes you think deeply about the behaviors I want to create. It helps me get into the mind-set on how to build these by thinking about the user interaction as well (P4)*

### 6.2.2 Ease of use for non-programmers:

The results from the numerical ratings that the authors provided on the difficulty level of using the system (shown in Figure 40) were also analyzed. Numerically, the authors provided a low rating on the difficulty level. On an average, authors provided a rating of 1.09 out of 5 on the difficulty level (0-not difficult at all and 5-very difficult). Based on the results of one-way ANOVA, the differences in ratings across the different author categories were not found to be statistically significant, $F(5,59)=0.29$, $p=0.917$.

Interview data further qualified the numerical rating. P10 supported the ease of use through his statement:

*I was able to create once I familiarized myself, I got the hang of it, writing the text was easy, combining all the actions was easy, after you get used to creating behaviors, it becomes easier, the more I got it, the easier it became and the more adventurous I got. (P10)*

Authors mentioned that the system was "not very technical and they need not have any programming experience to use the system" [P50]. They felt that once they "got over the initial learning curve, things became easier for them" [P6]. Authors mentioned that in just two hour session (the time of the user study session), they were "able to create a lot of interesting interactions" [P8]. P52 commented on the ease of use of the authoring process:

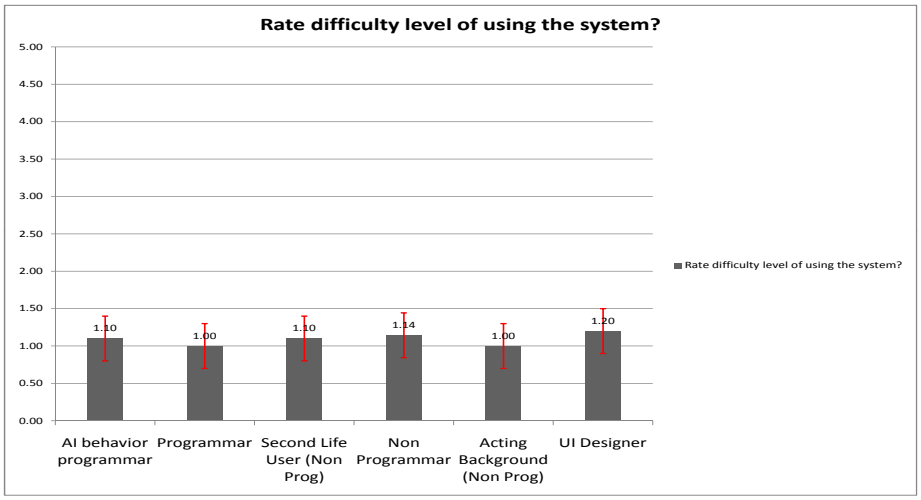



**Figure 40:** The figure shows the author's ratings on difficulty level of using the system. 0-not difficult at all, 5-very difficult

*The behavior creation was well defined with the different steps (triggers , actions etc). The skeleton of the system is very good and easy to understand. I liked the way I could add or edit triggers for the behaviors (P52)*

One of the key things that was planned to be tested through this phase was the difficulty level of using the authoring system. A non-programmer would not have the necessary background in terms of programming experience or AI expertise to use the system. The non-programmers should be able to create the behaviors and an overall experience that was as sophisticated as an expert in the domain (AI behavior programmers) using the authoring interface. Based on results from the interview data and numerical ratings provided by the authors (in Figure 40), it seemed that the non-programmers had an equal understanding of the system as an AI behavior programmer, an expert in the domain of behavior authoring.

In terms of the number of behaviors created, Post hoc comparisons using Scheffé test (shown in section B.3 in Table 11 of the Appendix) indicated that acting background non-programmers and programmers created the highest number of behaviors

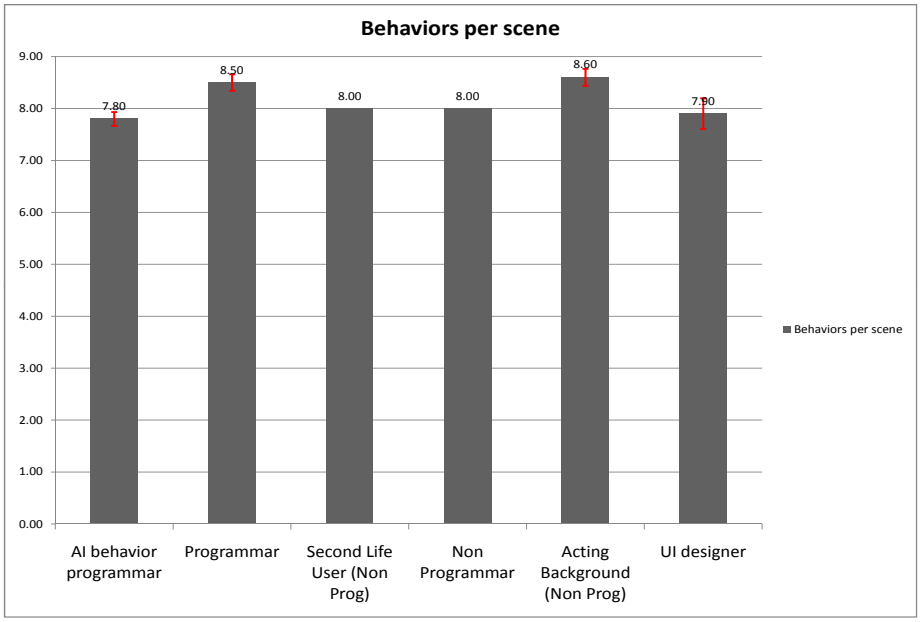



**Figure 41:** The figure shows total number of behaviors created by different authors

(8.6 and 8.5 on average respectively). Number of behaviors created by authors in other author categories were found to be statistically similar. The time spent on different authoring activities by the authors were also analyzed. I discuss them next.

**Trigger creation/correction time per behavior:** This measures the time spent by the authors in creating and correcting (after finding problems with them in Second Life) the triggers per behavior. During the authoring session, authors would correct the triggers after they view the behaviors in Second Life. Post-hoc comparisons using Scheffé test (shown in section B.3 in Table 11 of the Appendix) indicated that mean time spent by AI behavior programmers in creating and correcting their trigger sets was significantly different than authors in other author categories (shown in Figure 42. AI behavior programmers were found to spend the least amount of time creating as well as correcting the triggers for the behaviors (shown in Figure 42). As domain experts, AI behavior programmers seemed to have a better understanding of

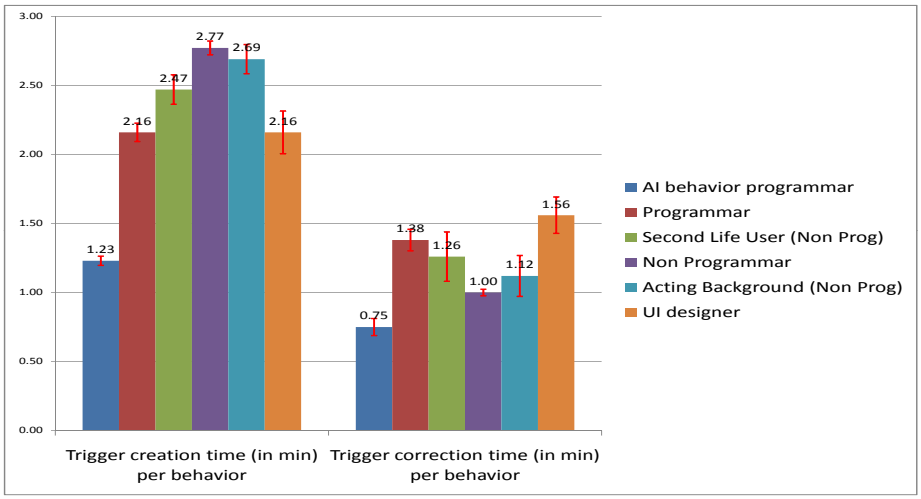



**Figure 42:** The figure shows time spent on trigger creation and correction by different authors. All the times are represented in minutes.

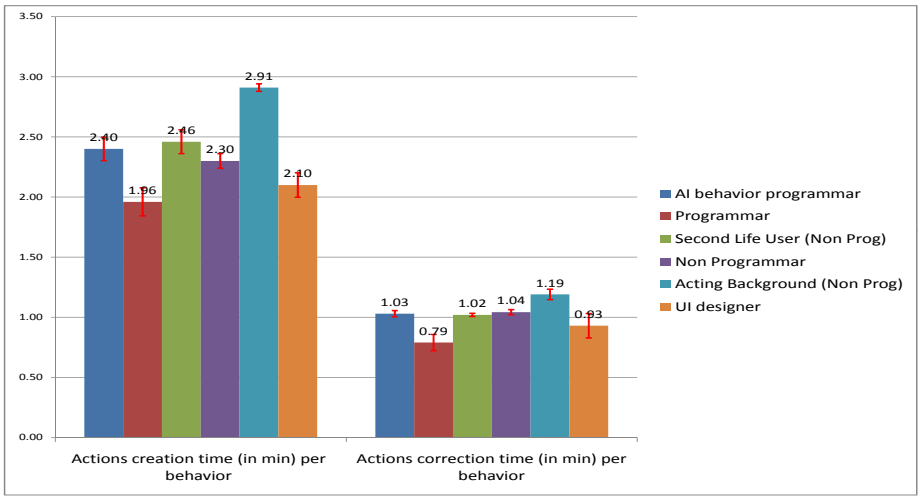



**Figure 43:** The figure shows time spent on action creation and correction by different authors. All the times are represented in minutes.

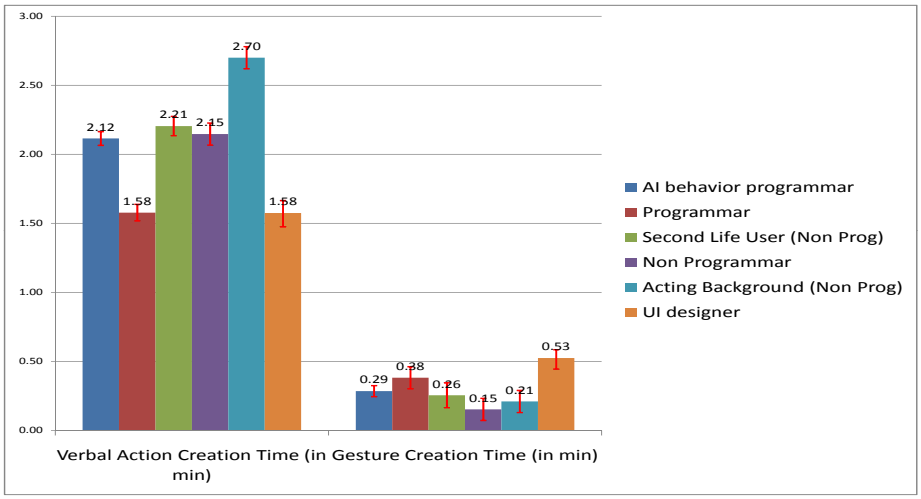



**Figure 44:** The figure shows time spent on gesture and verbal action creation by different authors. All the times are represented in minutes.

which triggers to use for the behaviors and that seemed to help them in creating the triggers in less amount of time compared to other authors as well as get the triggers right in the first place[1] which resulted in them spending the least amount of time correcting the triggers.

**Action creation/correction time per behavior:** This measures the time spent by the authors in creating and correcting the actions for the behaviors. During the authoring session, authors would correct the actions after they viewed and interacted with the authored behaviors in Second Life. Post-hoc comparisons using Scheffé test (shown in section B.3 in Table 11 of the Appendix) indicated that mean time spent by the acting background non-programmers in creating and correcting their action sets was significantly different than authors in other author categories (shown in Figure 43). The time spent by authors in other categories in creating and correcting the authored actions was found to be statistically similar (shown in Table

[1]As we will see from Phase III results that AI behavior programmers get the least amount of suggestions on their triggers because their triggers worked during player interactions

**Table 5:** The table shows overall time (in min) spent on a) trigger correction and correction and b) a) action creation and correction across different author categories. Category 1=Total time spent in trigger creation and correction, Category 2=Total time spent in action creation and correction, Category 3=overall time spent on authoring activity

| Cat. No. | AI beh. prog. | Prog. | Second Life Non Prog. | Non-prog. | Acting background | UI designer |
|---|---|---|---|---|---|---|
| 1. | 15.44 | 30.09 | 29.84 | 30.17 | 32.77 | 29.39 |
| 2. | 26.75 | 23.38 | 27.84 | 26.74 | 35.26 | 28.91 |
| 3. | 55.20 | 67.99 | 71.08 | 71.84 | 84.67 | 68.50 |

11 in section B.3 of the Appendix).

**Overall creation time:** Post-hoc comparisons using Scheffé test (shown in section B.3 in Table 11 of the Appendix) indicated that mean time spent by the Acting background non-programmers and AI behavior programmers in conducting the authoring activity was significantly different than authors in other author categories (shown in Figure 43. Acting background non-programmers spent the longest time conducting the authoring activity (shown in Figure 45) and AI behavior programmers the lowest amount of time. Authors in other author categories spent similar amount of time statistically. Analysis of the time spent on various authoring activities indicated that acting background non-programmers increase in time compared to other authors came due to the time they spent creating the actions (as shown in Table 5). For example, the total creation time difference between Non-programmer and Acting background programmers is (84.63-71.84=12.79) and is mainly contributed by the difference in action creation and correction (35.26-26.74= 8.52) which is 66% of the total creation time (shown in Table 5). However, this time was caused due to acting background non-programmers having highest number of verbal actions and gestures per behavior (shown in Figure 44) and the highest number of words per verbal action among the different author categories. The time taken by acting background non-programmers to create each word for a verbal action was however found to be

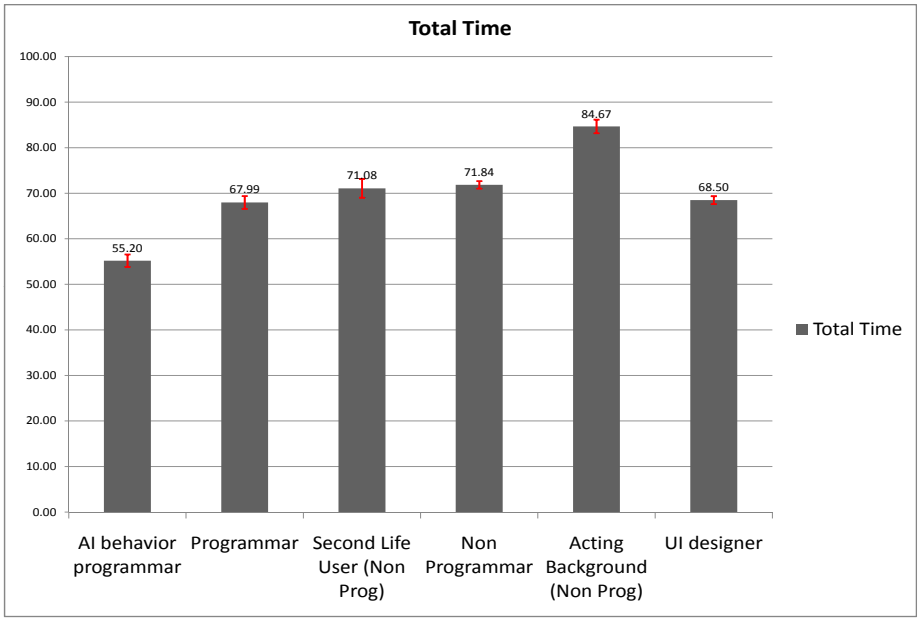



**Figure 45:** The figure shows total time spent on the authoring activity by different authors

the lowest (shown in Figure 46). AI behavior programmers spent the least amount of time compared to other authors due to least amount of time spent in creating and correcting the triggers (as shown in Table 5). For example, the difference in trigger creation and correction time between AI behavior programmer and non-programmer is (30.17-15.44=14.73) which accounts for 88.5% of the difference in overall creation time difference (71.84-55.20=16.64) between the two author categories.

**Actions per behavior:** Post hoc comparisons using Scheffé test (shown in section B.3 in Table 11 of the Appendix) indicated that mean number of gestures per behavior created by the Acting background non-programmers was significantly higher than authors in other author categories. Acting background non-programmers had the highest number of gestures per behavior on average. Gestures per behavior were found to be statistically similar across the other author categories (shown in Table 11 in section B.3 of the Appendix). Acting background non-programmers had the

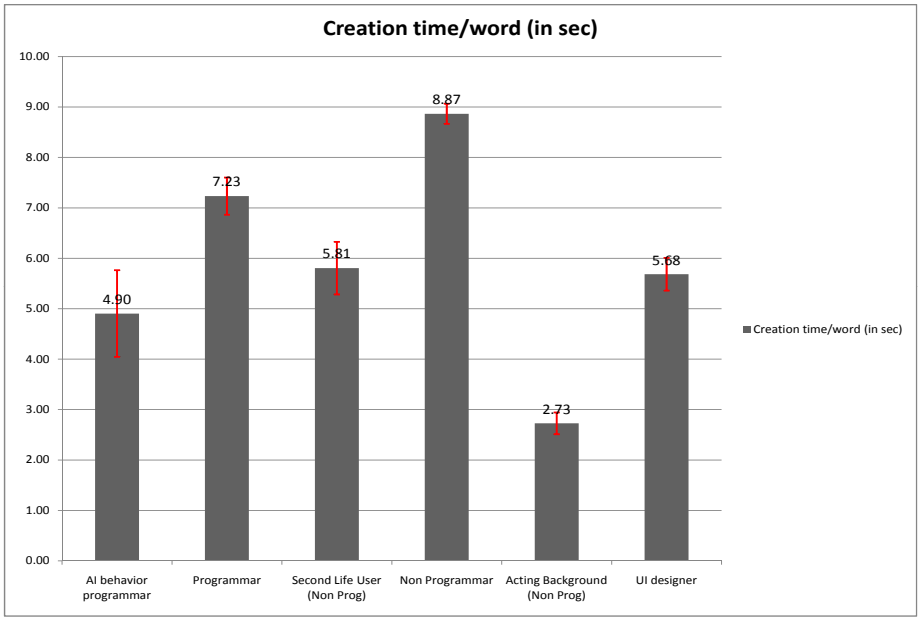



**Figure 46:** The figure shows time spent on creating a word for a verbal action by different authors

higher verbal actions per behaviors and programmers the lowest. Number of verbal actions per behavior across other author categories were found to be statistically similar (shown in Table 11 in section B.3 of the Appendix).

It remains to be seen how good the created behaviors are in terms of creating an interesting player experiences (evaluated through phase II).

**Timeline editor made the authoring process easier:** Authors mentioned that the possibility of adding things in the "order in which they will be executed (on the timeline in the head shaped editor) made the process very easy"[P5]. They felt that the step was the "easiest part and made it very easy to create behaviors" [P8]. The timeline view which "allowed for left to right sequencing was the easiest part and made it very easy to do things with a drag and drop ability" [P41]. P3 mentioned:

*The ability to detail the behaviors by just dragging any action and put it one after another was very convenient. I could drag any actions and could make my avatar*

*do anything-that made it very simple (P3).*

One of the design decisions that was taken for presenting the authoring task was to define the behavior creation task in a way that would be more closer to the way authors think about the creation process and would therefore help authors in conceptualizing the authoring process. As discussed earlier, based on the author feedback from the paper prototype study that they were more attuned to thinking of the behavior creation process as a sequential process, the notion of timelines, instead of a hierarchical organization was employed. Interviews with the authors revealed the ease of putting things in sequence. Author 33 mentioned:

*Being able to build things in the order in which you put them in, its very much you click it and its in, it came well to me, I was able to picture things and create them.(P33)*

### 6.2.3 Immediate real-time feedback helps find problems with the behaviors

The authoring system allowed authors to create behaviors in the Second Mind authoring interface, which they could then go and try out in Second Life. Authors mentioned that the "immediate feedback was very useful" [P45] and "gratifying"[P5]. They could go into the world and see the problems with the created behaviors like some action or trigger not working as expressed by P7:

*Thats the kind of thing that made it easy. So I made three things that made it difficult to happen at the same time and I realized that it couldnt happen that easily. So I could see that right away and you can alter that (P7)*

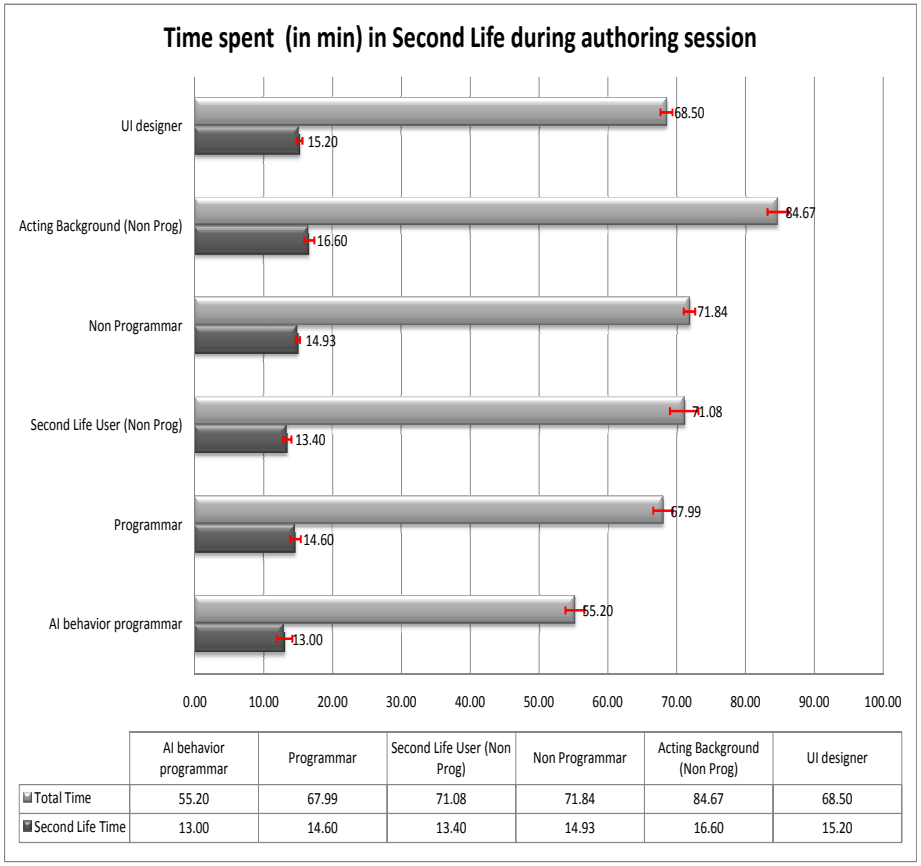
The ability to change the behaviors in Second Mind and go into Second Life and immediately view them without reloading the Second Life client was found to be very useful. P3 (an AI behavior programmer) expressed:

*The strongest point is the immediate feedback, I can try something and test it in realtime and see if that would work well. (P3)*

It is quite nearly impossible to take into account all the possible situations that might occur even for an expert AI behavior programmer. This task becomes harder for non-programmers as they are new in carrying out the task. Thus, it is very likely that while defining behaviors for believable characters, the author forgets to take into account certain situations. Authors comments from the interviews indicated that the integration with a concrete virtual world, provided them the ability to view the behaviors in action, execute them in a concrete virtual world and identify issues with them. During the authoring session, authors spent part of their time in debugging the behaviors where they went into the virtual world and identified some of the problems with the authored behaviors. P6 statement reflected this behavior:

*It was very helpful to go in second life and view it, twice I went into the world and see it didn't look like this, the animation looked like goofy and I thought this is not how I would like the avatar to act and then I modified it. (P6)*

Figure 47 shows the time spent by the authors in Second Life interacting with the avatars and testing the behaviors during the authoring session. On average 21% of the authoring session time was spent in Second Life interacting with the avatar and identifying problems with the created behavior sets. One of the problems that authors mentioned with the paper prototype was the inability to see the results of their creations. They were unable to identify the problems as they could not see a running simulations of their creations. Integration with Second Life allowed authors to see the authored behaviors in action, identify some problems with them and correct them as well.




| | AI behavior programmar | Programmar | Second Life User (Non Prog) | Non Programmar | Acting Background (Non Prog) | UI designer |
|---|---|---|---|---|---|---|
| Total Time | 55.20 | 67.99 | 71.08 | 71.84 | 84.67 | 68.50 |
| Second Life Time | 13.00 | 14.60 | 13.40 | 14.93 | 16.60 | 15.20 |

**Figure 47:** The figure shows the time spent by different author categories in Second Life interacting with their behaviors. Total authoring session time is also shown.

**Table 6:** The table shows results from behavior limitations across different author categories. Category 1=No. of behaviors (unable to author), 2=No. of behaviors (found a workaround), 3=Number of behaviors per scene 4=% behaviors unable to author after workaround

| Cat. No. | AI beh. prog. | Prog. | Second Life Non Prog. | Non-prog. | Acting back-ground | UI de-signer |
|---|---|---|---|---|---|---|
| 1. | 0.30 | 1.0 | 0.00 | 0.00 | 0.00 | 0.80 |
| 2. | 0.30 | 0.10 | 0.00 | 0.00 | 0.00 | 0.00 |
| 3. | 7.80 | 8.50 | 8.00 | 8.00 | 8.60 | 10.13 |
| 4. | 0.00% | 10.59% | 0.00% | 0.00% | 0.00% | 10.13% |

### 6.2.4 Behavior creation limitations

As discussed earlier, before the authors started using the authoring system, the authors were asked to sketch the behaviors that they would like to create for the avatar on a piece of paper. This was done so that the authors could think of the behaviors without knowing the limitations of the authoring system. Any behaviors that the authors were unable to create were marked and it was also noted if they were able to find a work-around to create that behavior. The analysis of the results are shown in Table 6. Programmers were unable to create 10.59% of the behaviors they had initially sketched out on the piece of the paper and UI designer 10.13% of their behaviors. AI behavior programmers were able to find work-around for the behaviors they were unable to create initially using the authoring interface.

Programmers mentioned that they would like to have a way to define behaviors that could be carried out randomly. They also mentioned a feature in the authoring interface that would allow them to assign probabilities to the behaviors. Programmers also expressed the need to view syntax so that they could change and modify things by going internally. These limitations reflected in a statistically significant lower rating (based on post hoc Scheffé test and shown in Figure 48) received from the programmers when they were asked to rate the system on how well it allowed them to create the scenes and behaviors they had in mind. This was also reflected in their

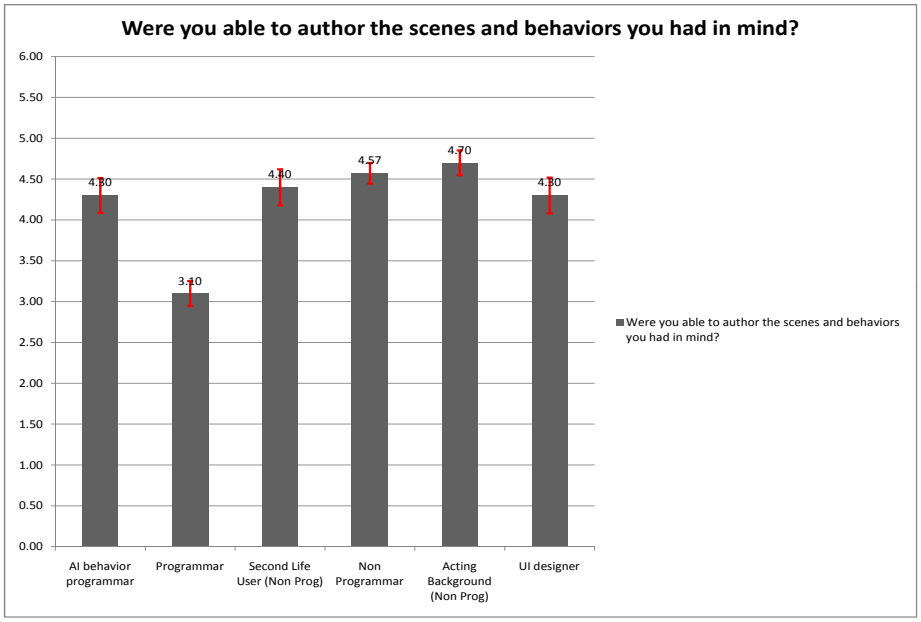



**Figure 48:** The figure shows ratings on whether the authors were able to create the scenes and behaviors they had in mind using the authoring interface.

lower rating on overall experience using the system (shown in Figure 49). Interestingly AI behavior programmers found the system adequate for the current authoring task (as expressed through a rating of 4.30). Programmers mentioned that they would like to have explicit support for behavior branching. Currently they had to do branching through a round about way. P7 mentioned:

*If I had to do any branching I had to have two different type of behaviors and two different type of triggers, which was a little round about way of doing it. (P7)*

**Action set limitations:** Some authors mentioned the need of having more animations that would allow them the possibility of creating various other types of behaviors. P11 mentioned:

*The visual buttons for the animations were very good, however, I would like to see more animations. If I had more quality animation then I might be able to create other*

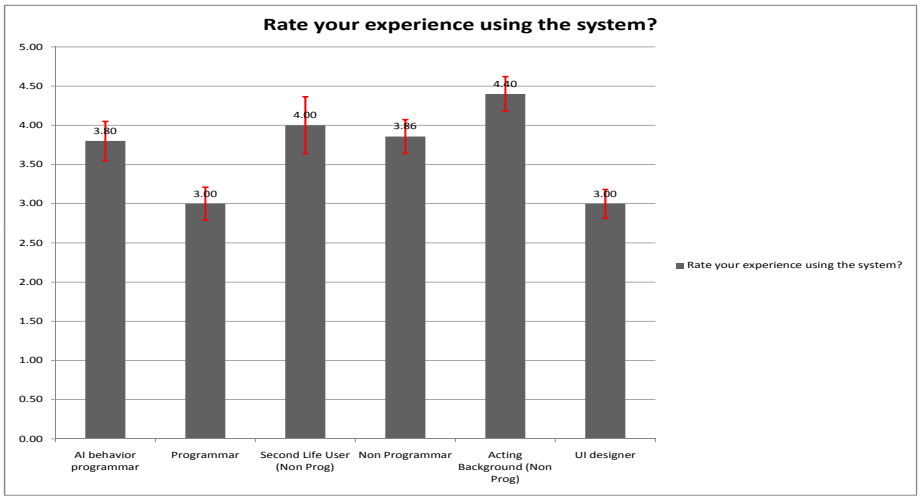



**Figure 49:** The figures shows the ratings by different authors on overall experience using the system

*different behaviors (P11)*

There wasn't any mention of trigger limitations during the interviews. Perhaps the kind of scenes (shopkeeper) they were asked to author didn't result in limitations on the triggers. Interviews also revealed certain other limitations in the authoring interface and suggestions on things that could be improved in the authoring interface. I present them next.

### 6.2.5 Other Improvements

- **Video example and tutorials:** The need for having video examples was mentioned by the authors. Authors felt that this would have "made the process easier for them" [P22]. Video examples that could have explained things to them in a step by step manner would have been more helpful than the text based tutorial. P34 mentioned:

  *To the text tutorial, I would add one video example with several behaviors that explains creating them step by step. (P34)*

- **Animation preview in Second Mind (authoring interface) itself:** Authors had difficulty understanding how the gestures would actually look like in the virtual world when selecting them in the behavior editor. In the current version, authors had to go in Second Life and see the animations to get a better understanding of how it looked on the avatar. Without the in-browser support, they were taking a guess (by looking at the names) on how the animation would look like and could not get a clear understanding of it.

  *If there were little examples of what the activity would look like for example, that would have been helpful. You could select wink or laugh and then have a preview type of thing to see the effect.(P10)*

- **Behavior linking with triggers:** Authors also mentioned the need of having a visualization of overall scene linkage with different behaviors and triggers that would give them an overall view of the created behaviors. P45 mentioned:

  *I would like to see a quick overview of the behaviors and what I had already done, all the time somewhere on the screen that I could follow, which ones I had already created and which ones I should add more to make this scene perfect. (P45)*

  As discussed earlier in Chapter 4 and showed in Figure 11, the behaviors defined for a scene are shown using a coverflow based approach. An extension of such a view which also shows triggers and that is also visible to the author at all times during the authoring process would help the author in getting a better overview of the behaviors that have already been created.

- **Better animation categorization:** Authors mentioned that it would be easier to select an action if there was a better categorization of the action list, instead of a long list (that is currently used, shown in Figure 50). If the actions were divided into for example, physical actions, emotions and verbal actions,

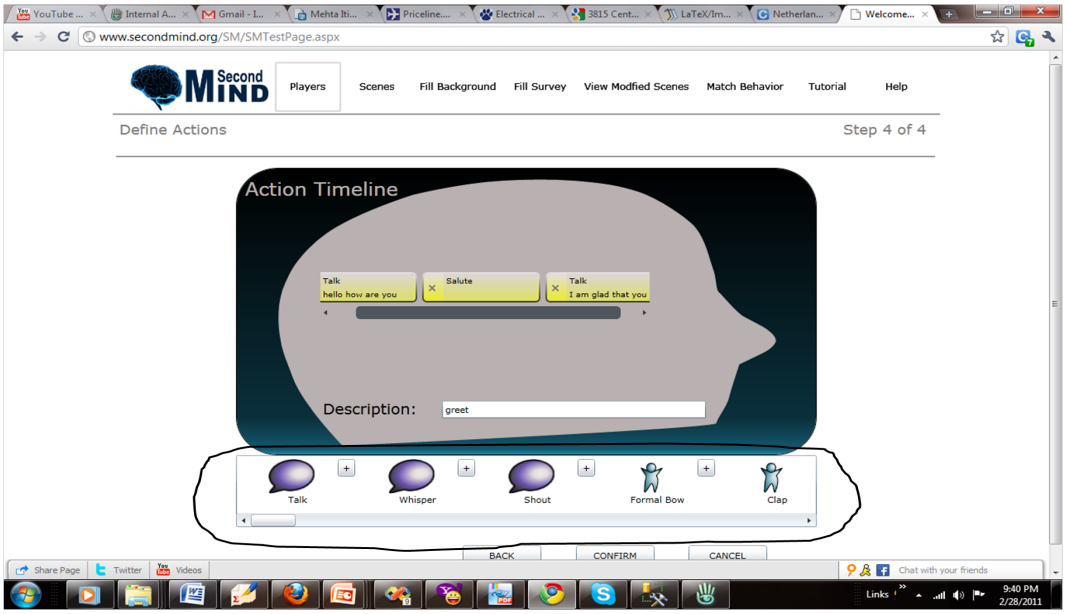



**Figure 50:** The figure shows the action sets available in Second Mind as a long list.

the selection process would have been easier. P49 mentioned:

*The thing I would suggest is to have a good division of animation, emotions and facial expression. That would allow for quick access. (P49)*

### 6.2.6 Summary: Phase I results

To summarize the results from the phase I study, authoring system provided non-programmers with programming scaffolding to author behaviors and interactions The authoring system provided support through three mechanisms a) Easy to use authoring environment: based on the numerical ratings and interview data, it can be concluded that the authors found that overall the system was easy to use. Non programmers found that they did not need any programming experience to use the system and provided numerical ratings (on ease of use) which were statistically similar in rating to AI behavior programming experts. Authors felt that using the timeline editor, the system allowed them to be very creative where they could go in the authoring interface and create in a small amount of time any kind of world that they had imagined, b) Understandable Vocabulary and Terminology: Non-programmers had

an equal understanding of the terminology on par with AI behavior programmers and c) Integration with the virtual world helped the authors identify problems with the behaviors: The authors could go in the virtual world and have a look at the created behaviors in the virtual world to identify any issues with the created behaviors.

The study also provided data on things that could be improved in the authoring interface. Authors mentioned that it would have been useful to have more video tutorials that they can just view instead of a text based tutorial file. Authors (programmers) wanted to have more advanced features like random behaviors or behaviors that could be selected based on probabilities. They also wanted to have features where they could go in and view the internals of the behaviors (like looking at the generated code) to modify it. Authors wanted to have an animation preview button where they could watch the avatar's animation right in the authoring interface itself (currently they had to go in second life and view the results of the animation they had selected). They also wanted the ability to create new animations to use them to create newer type of behaviors.

Once the behaviors were created by the authors, I wanted to find out how good the authored behaviors were in terms of creating a good player experience. Next, I discuss the results from Phase II.

## *6.3 Evaluation Study: Phase II*

Once the behaviors were created, another set of users (we use the term players for them) were invited to interact with the shopkeeper avatars (created as part of the first phase) in Second Life. Sixty five players were asked to interact with the avatars. Each player interacted with 15 avatars. As discussed earlier in section 4.2.4 of Chapter 4, the players were asked to fill out quantitative questionnaire that were dynamically generated based on the constraints associated with the behaviors that the avatars executed during the interaction. The players also provided feedback by answering

questionnaire on their overall experience (shown in section C.1 of the Appendix). This provided the necessary feedback for AI repair system to improve the behavior sets as well as gather the necessary ratings on performance of the authored behavior sets. The two questions related to the overall experience were:

Q=1 **Please rate your overall experience interacting with the avatar?**

Q=2 **Please rate avatar's overall performance acting as a shopkeeper?**

For a subset (around 5) of the players, written comments were collected on their experience after interacting with the avatars. These feedbacks were collected to get a bit more qualitative feedback on player's interaction with avatars created by different authors without doing an elaborate labor intensive qualitative analysis. One-way ANOVA analysis and post-hoc Scheffé tests (shown in section C.2 in Table 12 and section C.4 in Table 13 of the Appendix) were also conducted. Based on the quantitative ratings, authors who had an acting background received the highest rating on the two questions (shown in section C) (Q=1(M=3.71, SD=0.36), Q=2(M=3.61, SD=0.28)). Scheffé post-hoc test for significance across the different author categories were also conducted. The ratings for acting background were significantly higher when compared to the next closest category of authors (AI behavior programming background) Q=1(M=3.10, SD= 0.12), Q=2(M=3.20, SD=0.14). Scheffé Test statistic (Q=1 26.51, Q=2 12.07) for question 1 and 2 was found to be greater than the critical value (Q=1 11.85, Q=2 11.85) for p=0.05. Ratings for AI behavior programming background authors Q=1(M=3.10, SD= 0.12), Q=2(M=3.20, SD=0.14), Second Life background authors Q=1(M=3.07, SD=0.33), Q=2(M=3.03, SD=0.26) and Non-programmers Q=1(M=3.02, SD=0.21), Q=2(M=3.15, SD=0.25) were found to be statistically similar. There were significant difference between the ratings of non-programmers and UI designer as well. Scheffé Test statistic (Q=1 57.08, Q=2 62.99) for question 1 and 2 was found to be greater than the critical value (Q=1 11.85,

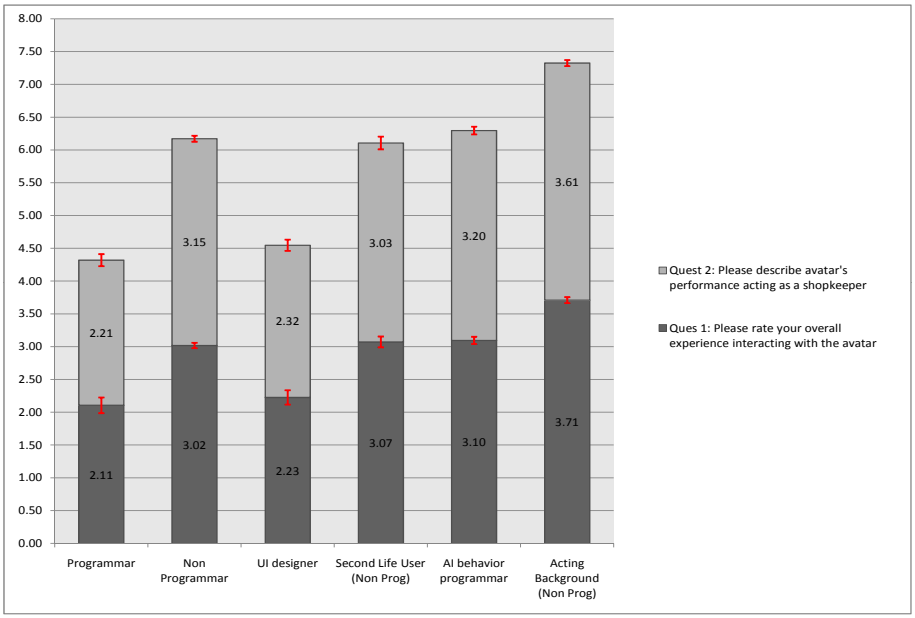



**Figure 51:** Ratings of authors behaviors in phase II

Q=2 11.85) for p=0.05. Ratings for UI designer and programmers were found to be statistically similar.

So to summarize the results. The ratings could be grouped into three categories:

I. **Acting background authors who got the highest rating.**

II. **Second life non-programmers, non-programmers with misc. background and AI behavior programmers who got statistically similar ratings.**

III. **UI designers and programmers who got the lowest rating.**

Once the ratings were obtained, I wanted to understand the reasons behind these differences. I specifically wanted to understand the following results:

1. **Non-programmers getting a rating that was similar to the rating received by AI behavior programmer who were domain experts.**

2. **Authors with acting background receiving the highest rating from the players.**

3. **Authors with programming background and UI designers receiving the lowest rating from the players.**

Interview data from phase I provided some answers to these questions. Player interaction from phase II and analysis of the authored behaviors also provided a better understanding of the reasons behind these differences. The analysis indicated the following:

- **Programming scaffolding helped the non-programmers:** The authoring interface provided non-programmers the necessary support through easy to use authoring interface, understandable vocabulary and concrete virtual world feedback on the authored behaviors allowing them to conduct the authoring process and create behavior sets that were either on par (in case of non-programmers with misc. background and Second Life non-programmers) or better (in case of acting background non-programmers) than AI behavior programmer created behavior sets.

- **Authoring interface supported author's creativity:** Authoring interface allowed the authors to detail the behaviors, express their creativity and use the design principles in authoring the behaviors.

- **Programmers and UI designers focussed on the limitations, unable to find work-around to create complete player experiences:** Programmers found that the authoring interface didn't provide them with advanced programming features to create behaviors. They were not able to work around these limitations (that other authors were able to) and use the existing authoring

interface to create complete player interactions. UI designers were similarly unable to work-around the usability limitations and other limitations like unable to modify the player GUI that he used to interact with the avatar.

Let's discuss these findings in detail next.

### 6.3.1 Programming scaffolding helped non-programmers

Phase I results supported the comparable player ratings of non-programmer authored behaviors with AI behavior programmer authored behaviors. Phase I results indicated that the authoring interface provided non-programmers the necessary programming scaffolding through a) Easy to use interface which allowed non-programmers to create the behaviors that they wanted to create, b) Understandable terminology that provided non-programmers an equal understanding of the vocabulary terms like (scenes, behaviors, triggers etc) as an AI behavior programmer who was an expert in the domain and c) Virtual world feedback that allowed them to create behaviors in the Second Mind authoring interface, which they were then able to try it out in a concrete virtual world, Second Life. The process of interacting with the authored behaviors helped them identify the problems with the created behaviors like some added action not working as well as they would have liked or a trigger not working properly. This programming scaffolding helped the non-programmers achieve player experience ratings on the scenes and behaviors that were higher (for acting background non-programmers) and ratings (for Second Life non-programmers and non-programmers with misc. background) that were statistically similar to that achieved by AI behavior programmers. With the necessary programming scaffolding, acting background authors were able to bring their creative skills and express themselves through the authoring interface.

### 6.3.2 Behavior detailing and creativity support

**Creativity support:** Authors mentioned that the authoring system provided them with the ability to create interactions (through creating scenes and behaviors for avatars) that they imagined themselves. The authoring system provided them the possibility of expressing their creativity, combining actions into behaviors in many different ways and according to some author's opinion with "limitless possibilities" [P2]. P17 supported this in his statement at the end of the authoring session:

> *I liked you could do it as a talk or shout which means you can get subtle with it and with little imagination it was not hard to come up with anything I would like to do. (P17)*

Authors mentioned that they could see themselves spending "hours creating worlds of their own imagination" [P5]. The following statement by author 6 expressed his opinion on expressivity of the authoring interface:

> *What I enjoyed when I was doing that was using my imagination to create this whole world that I visualize-this is what I want to see, this is how I choose things to be. This is technical but there is a creative aspect to it, I can create the kind of world I want to create. This allows you to use your creativity and go with it. (P6)*

**Behavior constraint support:** During the authoring process, authors defined constraints for the complete scene as well the behaviors. Assigning the constraints with the behaviors helped the authors think more deeply about the kind of player experience behaviors would create. Author 15 who was from the acting background mentioned:

> *Having to select constraints for the scene and the behaviors made me think more deeply about the player experience. It forced me to think about how the player would feel when interacting with the avatar. (P15)*

**Behavior detailing:** Authors mentioned that the ease of use of the editor provided them the ability to get adventurous with their avatars. They could go in and detail their characters by choosing the specific emotion and actions they wanted the avatar to do. P22 expressed:

*The ability to detail it, to put in the emotions, actions that I would like to have for the avatar, it has that ability. I could get very adventurous and choose things that I wanted my avatar to perform. (P22)*

Based on Pearson correlation analysis, time spent creating the actions indicated a high correlation value with player experience ratings (0.65). It would seem that detailing the characters by spending time in creating the actions was helpful in achieving higher player ratings. This careful detailing of the characters was highlighted by a statement from player 2:

*The interaction with this avatar (non-programmer authored) was much better (compared to programmer authored). The avatar was acting as a real salesperson and was very polite. There were good options to interact with the avatar. (Pl2)*

Pearson correlation analysis on the number of gestures and verbal actions per behavior with player experience ratings was also performed. The correlation with number of gestures and verbal actions (0.77 and 0.58 respectively) was pretty high. Figure 52 reflects this difference between a programmer and acting background non-programmer. Despite the high correlation, it would be presumptuous to say that having just the higher number of gestures and verbal actions per behavior would result in a better player experience rating. It would seem that having the right gestures in combination with carefully authored verbal actions would provide for a higher player rating as highlighted by comments from player 4:

*I had a better experience interacting with this avatar (created by acting background author). The dialogues and gestures that the avatar performed made a lot of sense*

*and worked well together. In some of the other avatars, it seemed avatar would do some animation but that wouldn't make sense sometimes. (PL4)*

**Design principles:** The authoring interface allowed authors to express and embed design principles as part of creating these characters. Authors with acting background mentioned employing Disney's character animation principle of "Exaggeration"[2] to create more believable behaviors. P20 (an acting background author) expressed this in his comments:

*We are in creating these characters exaggerating and in real life its one thing, you can get these emotions not being over the top, but here over the top is the way to go(P20)*

By providing authors who had a more creative bend, for example, author with the acting background with the necessary support through a) programming scaffolding, b) providing them with an authoring environment where they could go in and express their creativity and embed design principles, c) asking the authors to associate constraints with the behaviors that made them think more deeply about how the behaviors would create the right player experience and d) allowing them to detail the characters by choosing the right gestures in combination with carefully authored verbal actions, authors were able to achieve a higher rating (as seen in higher ratings for acting background authors). Player's comments after interacting with an avatar created by acting background non-programmers also reflected the creative nature of avatar's behaviors. Player 1 commented:

*Interacting with the avatar was very interesting (created by acting background). The avatar were very animated and made me laugh. They would come up with surprising*

[2]The definition of exaggeration used by Disney, was that perfect imitation of reality for cartoon characters would look dull and uninteresting. So in creating these characters, they would remain true to reality but be more wilder and more extreme in creating behaviors for these characters

**Table 7:** The table shows results from Pearson correlation coefficient for various quantitative measures in phase I with player experience ratings in phase II

| **Category Name.** | **Pearson Correlation Coefficient** |
|---|---|
| Gestures per behaviors | 0.77 |
| Verbal Actions per behaviors | 0.58 |
| Behaviors per scene | 0.27 |
| Trigger Creation Time | -0.44 |
| Action Creation Time | 0.65 |
| Second Life Time | 0.05 |
| Trigger Correction | -0.40 |
| Action correction per behavior | 0.31 |
| Total Authoring Time | 0.39 |
| Word per Verbal Talk action | 0.39 |

*behaviors during the interactions. He would come up with creative things to say. (PL1)*

Programmers and UI designers were however not able to achieve good ratings. Let's look into some of the possible reasons next.

### 6.3.3 Limitations for Programmers

As discussed in phase I results, programmers were unable to create 10.59% of the behaviors they had initially sketched out on the piece of the paper and UI designer 10.13% of their behaviors (shown in Table 6). Programmers wanted a way to define behaviors that could be carried out randomly and assign probabilities to the behaviors and were unable to create these behaviors or find a workaround for them using the current authoring interface. Programmers also expressed the need to view syntax so that they could change and modify things through the code. These limitations reflected in the lower ratings (shown in Figure 48) received from the programmers when they were asked to rate the system on how well it allowed them to create the scenes and behaviors they had in mind (shown in Figure 49). This was also reflected in their lower rating on overall experience using the system. Observations

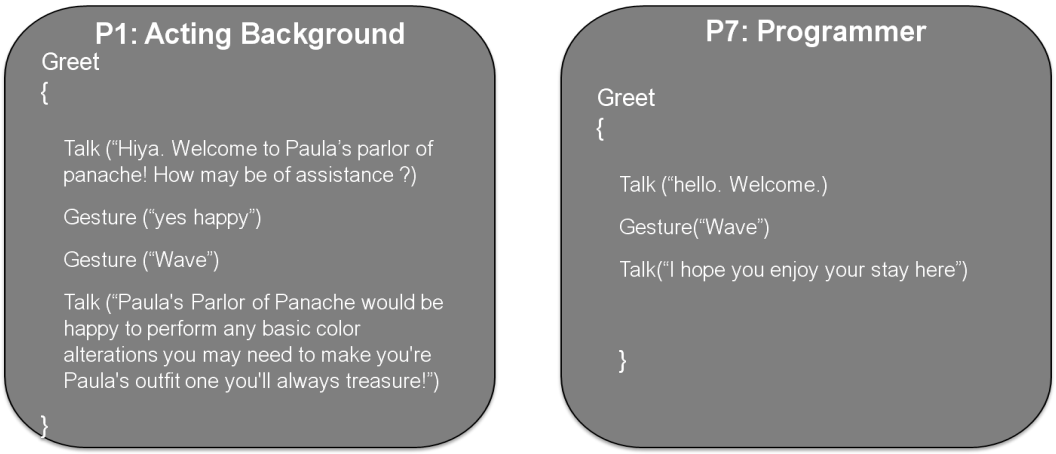



**Figure 52:** Difference between a greet behavior for a programmer and acting background nonprogrammer reflected by more gestures per behavior and more words per verbal action

of programmer's authoring session highlighted this as well. Programmer's spent time focussing on identifying (programming) limitations in the authoring interface. They were not able to find the necessary work-around to create behaviors they were unable to create due to limitations in the authoring interface. AI behavior programming authors (who had programming background as well), however were able to find the necessary work-around to create the behaviors they were unable to create due to authoring environment limitation. As shown in Table 6, AI behavior programmers were able to find work-around for 0.3 behaviors that they found they were unable to create due to authoring system limitations. This was also expressed by AI behavior programmers finding the system adequate for the current authoring task (as expressed through a rating of 4.30, shown in Figure 48). It is however unclear if programmers would have been able to achieve a higher rating if they would have not faced these limitations and if they had the advanced features in the authoring interface. Authors who received higher ratings than programmers had a better combination of gestures and verbal actions in their authored behaviors as well as more gestures and verbal actions compared to programmers. Figure 52 reflects this difference between a greet behavior of a programmer and an acting background non-programmer.

### 6.3.4 Usability limitations for UI designers

Observation of UI designer authoring session highlighted their focus on some of the usability problems with the system. P38 mentioned the usability problems he faced using the authoring interface:

*The usability part of the interaction flow is problematic, so it needs to be improved, like clicking on the button leads you to somewhere you don't know where you had reached, one button had different uses for example, you can't even use the browser back button, some parts of the interface work like flash and some like a web-site which causes a lot of confusion(P38).*

As discussed in phase I results, UI designers were unable to create 10.13% of their behaviors they had initially sketched out on the piece of the paper. UI designers wanted to create behaviors where the players would be allowed to provide richer sets of inputs into the system. UI designers wanted a way to change the current button based GUI so that they could allow the players to express their emotions (through emoticons for example) which the avatar could then react to. They also wanted a way to modify the layout of the button based player interaction GUI. P37 expressed:

*I didn't like the current layout of buttons. I wanted a way to modify it so that the player has a more friendly GUI to interact with (P37).*

The usability problems and the behavior creation problems they faced reflected in their lower overall ratings of using the system (shown in Figure 49). Observation of other author's authoring sessions as well as their ratings indicated that they were able to work around the usability limitations and use the authoring system to create the behaviors they had in mind. Other authors also reflected on some of the usability issues but seemed to familiarize with the system to create behaviors during the authoring session. P7 (acting background author) reflected this in his interview comments:

*Just familiarizing with it (the authoring interface), you click on the back button and it is not like a web-site and I was using a menu item like it was a back button in a web interface. When you have old things that are programmed, it takes some time to familiarize yourself(P7).*

Examples of player interactions with avatars created by programming background and acting background authors is highlighted in Figure 53 and 54 respectively. Table 7 also indicates Pearson correlation value for other quantitative measures in phase I with player experience ratings in phase II.

| | |
|---|---|
| **A(1):** | Gesture ("surprise")<br><br>Hiya. Welcome to Paula's parlor of panache! How may be of assistance.<br>Gesture ("yes happy"), Gesture ("wave")<br>Paula's Parlor of Panache would be happy to perform any basic color alterations or help you with any clothing needs you have. We will provide you an outfit that you'll always treasure<br>Gesture ("smile")<br>*(...user is presented with choices, "greet", "ask for assistance", "compliment")*<br>*(...silence for some time)* |
| **A(2):** | Let me know if you need any assistance! I am around and please feel free to ask anything.<br>Gesture ("smile"), Gesture ("bored") |
| **U(1):** | User clicks on button "ask for assistance". |
| **A(3):** | We have a nice collection that has come in.<br>Gesture ("point towards the player")<br>They are located on my right. I would be happy to show you the collection. Lots of people have really liked them.<br>Are you looking for something in particular today?<br>*(...user is presented with choices, "shirts", "trousers", "coat")* |
| **U(2):** | User clicks on the button "shirts". |
| **A(4):** | Some of the shirts are located on the right hand side? |
| **U(3):** | User goes close to the clothes closet. |
| **A(5):** | What would you like to choose? There are some new designer shirts as well as casual shirts collection that has come in.<br>*(...user is presented with choices, "white shirt", "designer shirts", "casual shirts")* |
| **U(4):** | User clicks on button "white shirt". |
| **A(6):** | Oh, this would be a great choice. I am sure you are going to like it.<br>Gesture ("Point You")<br>I have one of these as well that I bought for my brother-in-law. |
| **U(5):** | User clicks on button "purchase item". |
| **A(7):** | Excellent. I am glad you liked it.<br>This shirt has been one of our bestsellers for the last week or so.<br>I want to thank you for your purchase. This is going to look wonderful on you. |
| **U(6):** | User clicks on button "goodbye". |
| **A(8):** | Gesture ("wave")<br>Well, thanks for visiting Paula's Parlor of Panache. Hope you see you again soon!<br>I would get back to working at the register.<br>Gesture ("Wink")<br>Gesture ("toothsmile") |

**Figure 53:** Example of interaction from the phase II study. User interacts with the avatar created by an author with acting background

| | |
|---|---|
| | *(...user is presented with choices, "greet", "ask for help")* |
| **U(1):** | User clicks on button "greet". |
| **A(1):** | Welcome to my clothes store<br>Gesture ("laugh short") |
| **U(2):** | User clicks on button "ask for help". |
| **A(2):** | What specific clothes would you be interested in? Shirts? Trousers? Party dresses?<br>*(...user is presented with choices, "shirts", "trouser", "party dresses")* |
| **U(3):** | User clicks on button "party dresses". |
| **A(3):** | The specific thing you are interested is in that isle. the items are priced based on warranty and quality.<br>Gesture ("point towards the player") |
| **A(4):** | What price point are you trying to limit this purchase to?<br>*(...user is presented with choices, "100-200", "200-500", "500 and above/party dresses")* |
| **U(4):** | User clicks on button "500 and above/party dresses". |
| **A(5):** | As you are looking for party dresses, they are on the right section of the shop. Please come with me and I'll show you the section. As you already know any item which is pink in color would have extra 50% discount. Also the dress section has additional sale which is buy one and get one free. Its a great offer and there has been huge rush for this offer. I would also like to tell you that these offers are just for today so you might not see these tomorrow. Its a great deal!.<br>Gesture ("laugh")<br>*(...user is presented with choices, "purchase party dress", "look for something else", "dislike dress")* |
| **U(5):** | User clicks on button "purchase party dress". |
| **A(6):** | Oh these are great items which you've selected to buy! You are almost getting these at peanut prices! I'll do the billing!.<br>Gesture ("Surprise")<br>*(...user is presented with choices, "goodbye", "insult shopkeeper")* |
| **U(6):** | User clicks on button "goodbye". |
| **A(7):** | Gesture ("wave")<br>Thank you for shopping with us! I hope to see you again soon! Good-Bye. |

**Figure 54:** Example of interaction from the phase II study. User interacts with the avatar created by an author with programming background

## *6.4 Evaluation Section: Phase III*

Once players had interacted with the authored behaviors in phase II, I wanted to evaluate the repair approach. Thirty authors were invited back from phase I, 10 AI behavior programmers, 10 Second life experts and 10 Acting background authors. The selection of these author categories were done for the following reasons:

- AI behavior programmers were picked for phase III study as they were the domain experts and I wanted to find out if they would find the repair system helpful.

- Acting background non-programmers had the highest rating even without the repair system suggestions. Therefore I wanted to find out if repair system would still be able to provide useful suggestions and help them improve on their existing player ratings.

- As Second Life non-programmers and non-programmers with misc. background had similar ratings, I decided to pick one (i.e. the Second Life non-programmers). It was also guided by the logistical issue of being unable to get sufficient number of non-programmers with misc. background authors back for another study.

- UI designers were excluded from Phase III study as UI designers wanted different type of support and enhancements from the authoring interface As discussed earlier, UI designers wanted to create behaviors where the players would be allowed to provide richer sets of inputs into the system. UI designers wanted a way to change the current button based GUI so that they could players to express their emotions (through emoticons for example) which the avatar could then react to. They also wanted a way to modify the layout of the button based player interaction GUI. The repair system would not have been able to provide these enhancements to the UI designers. As the focus of this study

was on the repair system and kind of support it can provide to the authors, UI designers were excluded from this study. The programmers were excluded from Phase III study as programmers wanted to have programming features like the ability to define behaviors that could be carried out randomly and assign probabilities to the behaviors. Repair system would have still helped improve their behaviors as their behaviors also suffered from similar issues as behaviors from other category authors, however, they also wanted other features that the repair system would not have been able to provide and needed some other modifications in the authoring interface that was not the current focus.

During this phase, in the first step, the authors were presented with text based feedback that highlighted how well the behavior sets performed in terms of player ratings. The text based feedback provided simple textual descriptions to the authors detailing the performance of the behaviors with respect to associated constraints. The text based feedback was based on player feedback provided in phase II. An example of the text based feedback that was presented to the authors is the following:

*You have received a score of 1 for the constraint enthusiastic. You had defined a constraint value of 5. You need to modify the behavior to better satisfy user's expectations.*

In the second step, the authors were presented with the suggestions produced by the AI repair system. Authors were told that they could modify the behaviors further if they would like to add/change anything on those suggestions. As the authors started using the suggestions in the authoring interface, they were also asked to talk aloud about any modifications/additions they were conducting and their thoughts on the produced suggestions. These were noted down and discussed during the open-ended interview session.

Authors went through each suggestion and rated them on a 0-5 point scale. Authors were asked to fill out a quantitative questionnaire (shown in section D.1 of the Appendix) at the end of their session. They were also interviewed at the end of their session using the questionnaire shown in Figure D.1. In order to analyze the data, a grounded theory based approach similar to the one carried out for phase I data analysis was conducted. One-way ANOVA analysis and post-hoc Scheffé tests (shown in section D.2 in Table 16 and section D.4 in Table 17 of the Appendix) were also conducted. The analysis provided with the following results:

- **Design scaffolding through addition suggestions**: AI addition suggestions provided the necessary design scaffolding for authors.
  - The addition suggestions helped complete the missing interactions within a scene. The addition suggestions generated by the AI repair system also helped the authors by giving them new ideas on additional behaviors that could be added to the scenes.
- **Debugging Scaffolding through trigger suggestions:** AI trigger suggestions helped the authors in identifying problems with the behaviors and correct them.
  - The trigger suggestions generated by the AI repair system were very helpful (esp. for non-programmers) in finding problems with the behaviors.
- **Numerical ratings were helpful but didn't tell "how" to modify the behaviors:** The numerical ratings provided in the first step were found to be helpful to some extent by the authors. The numerical rating provided authors with an idea of player's perception of the behaviors and the authored scene. Although numerical ratings were helpful, they did not however, provide authors with an idea on what was missing from the behaviors and how they could modify

the behaviors to achieve better ratings.

- **Limited justifications on AI suggestions**: AI suggestions were found to be limited in terms of providing the justifications on why an AI suggestion was being made.
  - The suggestions produced by AI repair system were found to be limited in providing detailed reasoning on "WHY" an AI repair suggestion was made. This missing reasoning explanations would have been helpful in achieving more buy-in from the authors.

In the next section, I detail on these results.

### 6.4.1 Numerical Ratings were helpful to some extent

The numerical ratings that were presented to the authors in Step 1 of Phase III were found to be helpful to some extent by the authors. The numerical rating provided authors with an idea of player's perception of the created behaviors and the authored scene. The ratings provided authors with an idea of "aspects that were missing from their behaviors" [P16]. Numerical rating although not perfect helped give them some idea of behavior's performance and where things needed to be done further in order to improve the behavior's rating. P24 mentioned:

*I think there is a reason when you go to a hospital, they ask you to rate the pain on a scale of 0-10. We don't have a scale in our bodies and don't have numbers in our bodies that tell us exactly the numerical value, but it helps in having us a range, give us an idea. Its the same thing with these numerical ratings, by having a numerical scale, it allows you to see what I have got here. Its supposed to be a 4 and I only got a 2, so what I got to do is to make it twice as enthusiastic as it was before and if I could have gotten this far with my authored behavior I could plug in a few more things and get it further along. (P24)*

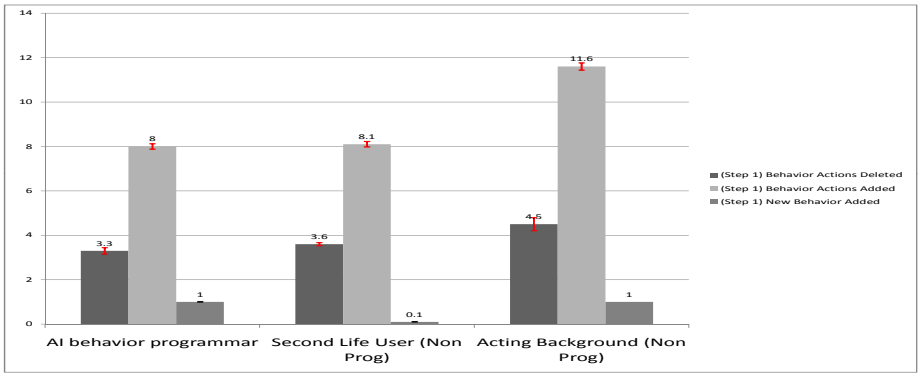



**Figure 55:** The figure shows breakdown of authors addition, deletions and new behaviors added in Step 1 of Phase III


**Figure 56:** The figure shows the modifications performed by the author himself after looking at the numerical rating

An example of a modification carried out by the author after looking at the numerical rating is shown in Figure 56. The overall action additions, deletions and new behavior additions across the various author categories are shown in Figure 55. The numerical ratings were found to be in general useful but it was hard for users to find out what was missing from the behaviors. The ratings provided authors with a sense of which action sets would provide a better player experience and they can use these highly rated actions sets to get them a higher score.

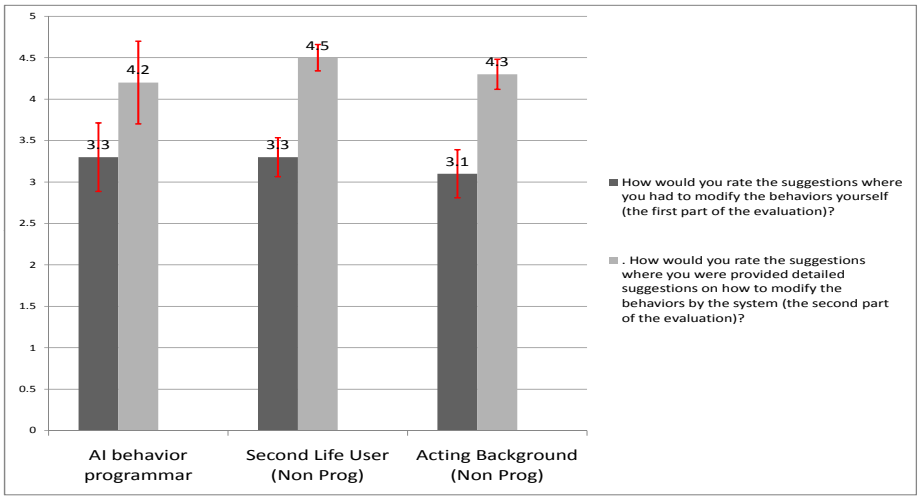



**Figure 57:** The figure shows overall ratings on numerical suggestions and AI repair suggestions

The numerical rating, however, had some limitations. The numerical ratings did not provide authors with ideas on "HOW" they should go about modifying the behaviors when they found out that the numerical ratings provided by the players were way off compared to what they had assigned during the authoring phase. The ratings did not provide them with an idea of "WHY" the users did not like their behaviors. P2 mentioned this limitation in his comments:

*I still have no idea of how I should modify the behaviors to match the user's expectation. The best I can do in this case is to try to change my behaviors to try to match user's expectations. (P2)*

The lack of detailed suggestion made the authors make guesses on ways in which they could modify the behaviors. P14 mentioned:

*The feedback (numerical rating) was helpful. I got to see which behaviors were close to the ideal and which weren't. It couldn't show me which direction to go in. So sometimes the feedback just told me I was off and didn't give a suggestion of which*

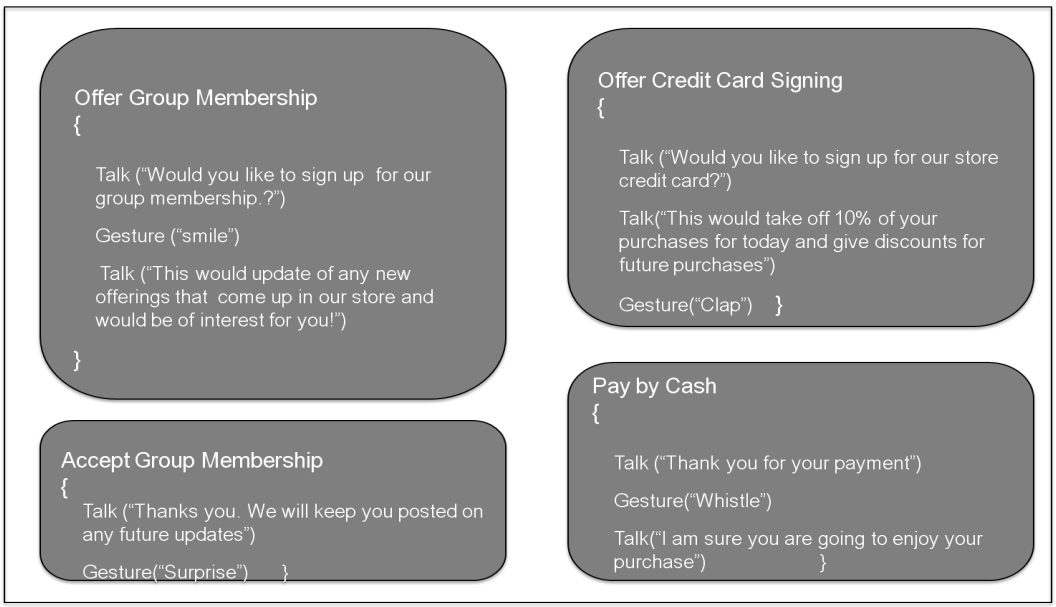



**Figure 58:** The figure shows new behavior additions that are carried out by the author after behavior payment by credit card is suggested

*direction to go, but I could usually guess which would be an appropriate direction to take. (P14)*

The AI repair suggestions provided in the second step of phase III helped the authors by providing directions on how behaviors could be modified to achieve a better rating. The limitations of the numerical ratings was overcome by the suggestions provided by the AI repair system (reflected in the difference in ratings as shown in Figure 57). P15 mentioned:

*The feedback overall was definitely helpful, may be not so much when I had to modify the behaviors myself (by looking at the numerical rating). Initially when it said that I have received a rating of 3 for a given behavior, and was asked to update it, it was hard to think of ways in which the behavior could have been improved. It made much more sense when I saw the suggestions in the second part of the evaluation (through the repair system (P15)).*

### 6.4.2 Repair suggestions helped in providing new ideas

The AI repair suggestions provided authors with new ideas on things that could be added to the behaviors. The suggestions at times, would provide them with another "IDEA" of how they could change their scene or other new behaviors they could add or things they could modify to change their existing behaviors. P22 suggested this benefit of the addition suggestion in his statement:

> *It was great to have these addition suggestions, even though sometimes if I wouldn't use them, it would give me an idea of what other new behaviors to add, like you are brainstorming with ideas that you might not use directly but would be useful in providing newer ideas(P22)*

Authors felt that as a first time user this feature was "very helpful"[P10] for them. Other authors who would be using the system for the first time "will have similar problems if not the same and they will find the suggested fixes very useful"[P30]. Authors felt that the authoring system felt like a collaborative system where "some parts were being done by the computer and some by the author and both were helping each other to get things done[P28]." P30 expressed:

> *I think for sure it[addition/brainstorming] was very helpful. Computer adding new actions helps the user think of new actions themselves. There is no way to avoid new inspirations by throwing new inspirations [through AI repair suggestions] at them. (P30)*

During the authoring process, authors missed some behaviors that needed to be part of the scene to complete the interaction for that scene. Examples of these include, for example, like a payment method for the shopkeeper scene, which would have allowed the players to pay for the purchased item. Addition suggestions were helpful in filling out these holes. P23 mentioned:

*When I am building the sales process, then I am doing it for the first time so there are going to be holes and it (addition suggestions) feels to me that this is like filling in the holes. (P23)*

An example of a behavior that was suggested as addition is payment by credit card:

```
Pay by Credit Card
{
    Talk ("Thank you for your purchase sir. This is wonderful.
           I am glad you came and visited us today")
    Gesture("Toothsmile")
    Talk ("I think you will find this purchase very useful and
           would give us another opportunity to serve you in
           the future")
}
```

Figure 58 shows new behaviors like "offer group membership", "offer credit card signing", "pay by cash" and "accept group membership" that were added by the author when suggestion "pay by credit card" behavior is made to the author.

The average number of behaviors added across the different author categories as a result of addition suggestions are shown in Figure 59. Addition suggestions were found to be extremely helpful for AI behavior programmers and Second Life non-programmers. Post-hoc comparisons using Scheffé test (shown in section D.4 in Table 17) indicated that mean number of behaviors added by the AI behavior programmers was statistically similar to Second Life non-programmers and significantly different from Acting background non-programmer. Based on the addition suggestions, AI behavior programmers and Second Life non-programmers added 3.25 new behaviors on average (as shown in Figure 59). The AI addition suggestions provided authors

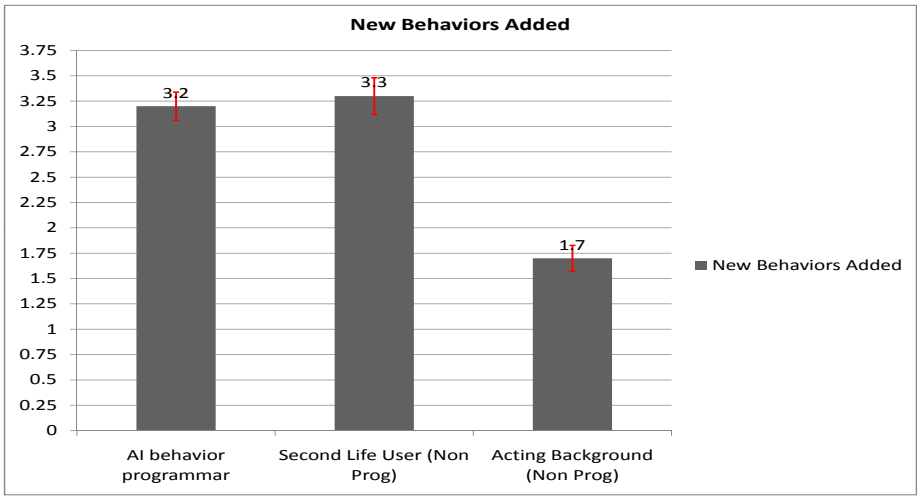



**Figure 59:** The figure shown the average number of behaviors added across different author categories after addition suggestions

with the necessary design scaffolding by opening up new behavior design spaces. An addition suggestion would provide them with new ideas on what other interactions (in the form of new behaviors) could be added to the existing behavior sets to create a richer player interaction.

As discussed in section 6.2.2, acting background non-programmers spent the longest time conducting the authoring activity (shown in Figure 45) and had the higher number of behaviors (8.6) as part of creating the scene. Acting background non-programmers also received the highest number of ratings from the player experience perspective in phase II. As acting background non-programmers had a better behavior set, they got lower number of suggestions (1.9) (shown in Figure 60). AI behavior programmers and Second Life non-programmers, on average got 3 addition suggestions. Post-hoc comparisons using Scheffé test indicated that mean number of behaviors suggested to the AI behavior programmers was statistically similar to Second Life non-programmers and significantly different from Acting background non-programmer. The figure also shows the breakdown of additions that were accepted

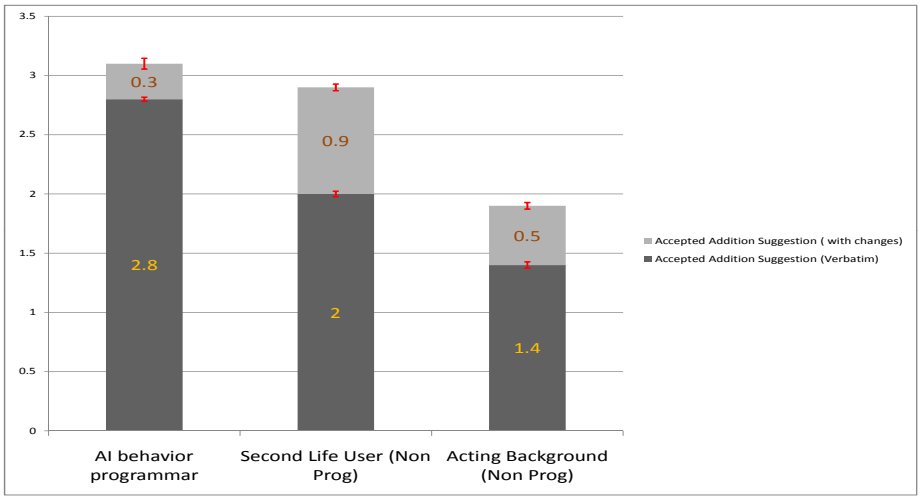



**Figure 60:** The figure shows breakdown of authors modifications of addition suggestions in Step 2 of Phase III

verbatim and those that had some modifications performed by the authors. On an average about 78.5% of the AI addition suggestions were accepted verbatim.

### 6.4.3 Trigger suggestions were very helpful

Authors were presented with modification suggestions on the existing triggers that they had created during phase I authoring. The suggestions provided a way for authors to see why some of their behaviors could not have been triggered in phase I. P2 described the problems he faced with the triggers not working properly during phase I:

> *I liked that a lot (trigger suggestions), Because I remember when I did it the first time some of my triggers didnt even happen. It (repair system) suggested that I fix those and then they will happen, that was very helpful (P2).*

Trigger suggestion were found to be especially helpful for non-programmers. post-hoc comparisons using Scheffé test (shown in section D.4 in Table 17) indicated that mean number of triggers suggested to the Acting background non-programmers and

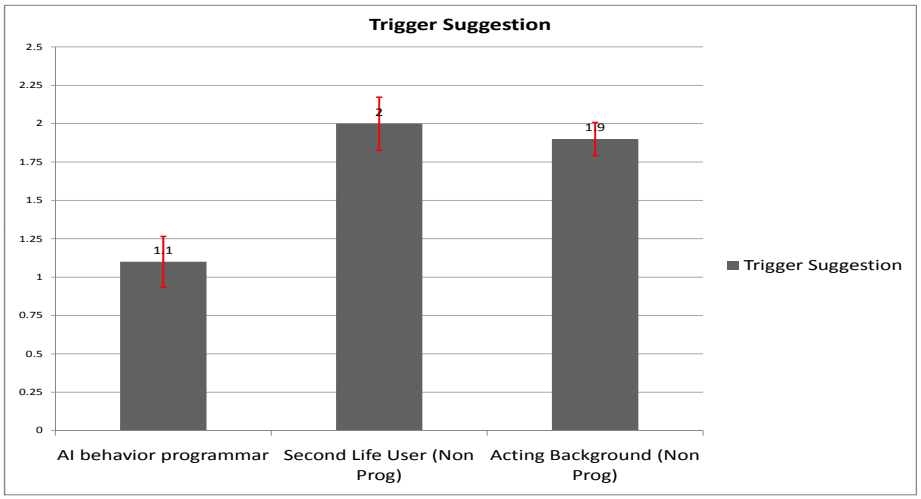



**Figure 61:** The figure shows the trigger suggestions that were produced for different author categories

Second Life non-programmers was statistically similar and significantly different from AI behavior programmers. On an average, AI repair system produced 1.95 trigger suggestions for non-programmers compared to 1 trigger suggestion for AI behavior programmers (shown in Table 61). As it has been seen in section 6.2.2, AI behavior programmers spent the least amount of time creating as well as correcting the triggers for the behaviors. As domain experts, AI behavior programmers seemed to have a better understanding of which triggers to use for the behaviors and that seemed to help them in creating the triggers in less amount of time compared to other authors as well as get the triggers right in the first place which resulted in them spending the least amount of time correcting the triggers. As a result, the AI behavior programmers got the least amount of suggestions on their triggers because their triggers seemed to work much better during player interactions.

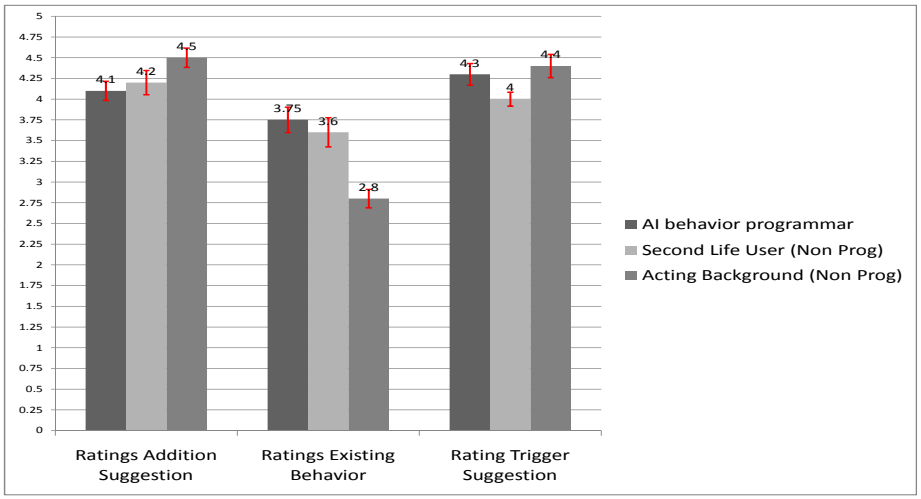



**Figure 62:** Ratings across various author categories on various different suggestions produced in phase III

### 6.4.4 Missing Detailed Explanation on "why" AI repair suggestion were picked

As discussed in section 5.3.3.4, authors were provided reasoning on why an AI repair suggestion was being made. Authors felt that these explanations were limited and they would like to have more in depth explanation on why a particular suggestion was being presented to them and why adding it would help improve the rating as expressed by P11:

> *I would like to know the reasons on why I was being suggested a new behavior. The current reasons (that were given) were helpful to some extent but I would like some more detailed explanation. (P2)*

Authors mentioned the need for improving the explanation behind the presented suggestion. This would help them understand more of the reasoning behind the changes and would also be helpful in achieving more buy-in from the authors (reflected in a little lower ratings on suggestions provided by the authors on existing behaviors, shown in Figure 62). P13 mentioned:

**Table 8:** The table shows some of the codes used for data analysis of phase III

| **Category** | Player Comments |
|---|---|
| **AI addition suggestion help in brainstorming** | It was great to have these addition suggestions, it gave me an idea of new behaviors that could be added. It felt like you are brainstorming new ideas. |
| **AI suggestions help fixing behaviors** | The suggestions fixed things by itself and I don't have to do much, I don't have to think too hard, that was quite useful. |
| **New Suggestions provided new ideas** | Computer adding new actions would help the user think of new actions and if not it will still enrich the experience. There is no way to avoid new inspirations by throwing new inspiration at them. . |
| **Feels AI suggestions would result in more satisfied customer** | In most cases, they provided actions that made a lot of sense and would probably result in a more satisfied customer. |
| **Trigger Suggestion was very helpful** | I remember when I did it the first time some of my triggers didn't even happen. It suggested that I fix those and they will happen, that was very useful . |
| **Feels numerical rating was helpful** | It was helpful as it gave me definite idea on what was way off, It was useful for sure. |
| **Wants to have detailed reasoning for AI suggestions** | Yes, having a "why" would have been helpful. It would help me see why some suggestions were made by the computer. |

*Yes, having a why something is suggested needs improvement. I would say what was missing was the semantic reason behind the changes? (P10)*

## *6.5 Results Summary: Phase III*

Results from the phase III indicated that the AI repair suggestions were able to provide two kind of scaffoldings, a) Design scaffolding in the form of addition suggestions that helped the authors by providing them with new ideas on behaviors that could be added to the scenes and b) Debugging scaffolding in the form of trigger suggestions that were found to be very helpful (esp. for non-programmers) in finding problems with the behaviors. The numerical ratings provided in the first step were also found to be helpful to some extent by the authors but that did not however, provide authors

with an idea on what was missing from the behaviors and how they could modify the behaviors to achieve better ratings. The suggestions produced by AI repair system were found to be limiting in providing detailed reasoning on "WHY" an AI repair suggestion was made. This missing reasoning explanations would have been helpful in achieving more buy-in from the authors.

Once the behaviors were repaired, I wanted to find out how good the repaired behaviors were in terms of creating a good player experience. Next, I discuss results from Phase IV.

## *6.6 Evaluation Section: Phase IV*

In the fourth phase, another set of players interacted with four sets of behaviors. The first set was the behaviors that were originally created by the authors in phase I (termed original behaviors). The second set of behaviors were the ones adapted by the authors using the numerical rating shown to them in Phase III (termed author adapted). The third set of behaviors were the ones that have been modified based on AI repair suggestions (without any author intervention)(termed AI adapted). The fourth set of behaviors were the ones that have been modified in collaboration between AI and the authors (termed Mixed adapted). Players interacted with these four behavior sets in Second Life and provided feedback on their interaction. The ratings on these behavior sets provided a way of measuring and comparing AI repair approach effectiveness in appropriately adapting the behavior sets and achieving a better player experience. Fifteen players were asked to interact with the thirty avatars. Each player interacted with 30 avatars. As discussed earlier in section 4.2.4 of Chapter 4, the players were asked to fill out quantitative questionnaire that were dynamically generated based on the constraints associated with the behaviors that the avatars executed during the interaction. The players also provided feedback by answering questionnaire on their overall experience (shown in section C).

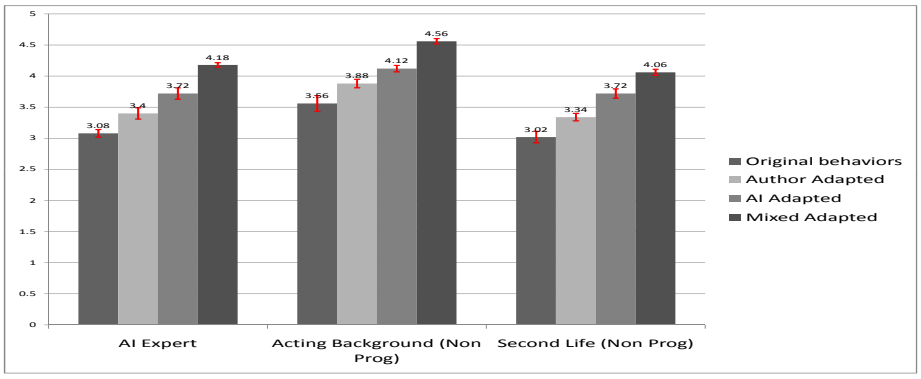



**Figure 63:** The figure shows player's ratings for different author category behaviors in phase IV for question 1

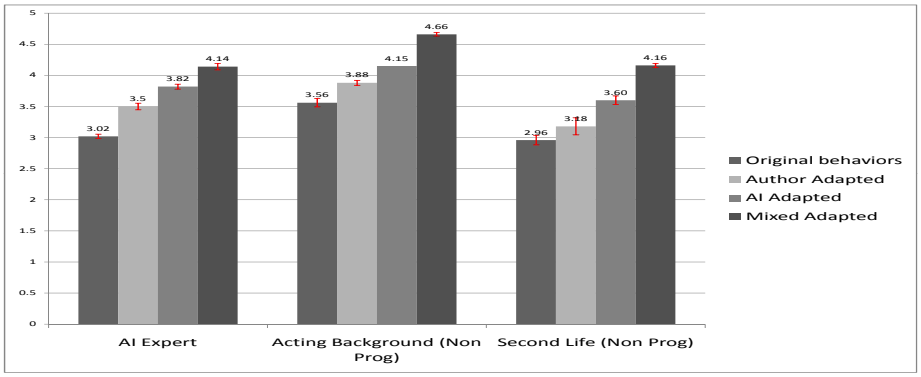



**Figure 64:** The figure shows player's ratings for different author category behaviors in phase IV for question 2

I conducted one-way ANOVA analysis and post-hoc Scheffé tests (shown in section E.1 in Table 18 and section E.4 in Table 20 of the Appendix). I ran Scheffé post-hoc test for significance for different behavior sets for different author categories. For acting background non-programmers, the difference in ratings between a) original and author adapted, b) author adapted and AI adapted, and c) AI adapted and Mixed adapted were found to be statistically significant. For Second Life non-programmers and AI behavior programming authors, the difference in ratings between a), b) and c) were also found to be statistically significant. The two questions related to the overall experience which the authors answered were the same as for phase II:

Q=1 **Please rate your overall experience interacting with the avatar?**

Q=2 **Please rate avatar's overall performance acting as a shopkeeper?**

The ratings for the two questions are shown in Figure 63 and 64 respectively. To summarize the results, the ratings could be understood in the following ways:

**Mixed adapted behavior sets across the different author categories got the highest player rating followed by AI adapted, author adapted and original behavior sets.**

Once the ratings were obtained, I also wanted to understand the reasons behind these differences. I specifically wanted to understand the following things:

1. **The mixed adapted behavior sets getting a rating that was higher than the rating received by AI adapted behavior sets.**
2. **The AI adapted behavior sets getting a rating that was higher than the rating received by author adapted behavior sets.**
3. **The author adapted behavior sets getting a rating that was higher than the rating received by original behavior sets.**

Interview data from phase III provided some answers to these questions. Player interaction from phase IV and analysis of the authored behaviors also provided a better understanding of the reasons behind these differences. The analysis indicated the following:

- **Testing scaffolding** in the form of numerical feedback from players helped authors understand the behaviors that were not working well and helped them adapt the behaviors.
- **Debugging scaffolding** in the form of feedback on triggers and existing behaviors led to AI adapted behaviors receiving a higher rating than author adapted behavior sets
- **Design scaffolding** in the form of addition suggestions that provided authors with new ideas on things that could be added to the behavior sets in order to create a more richer interaction.

I discuss these in detail next.

### 6.6.1 Testing scaffolding

The authoring system provided authors with the framework where they could get numerical feedback from the players. Authors would assign constraints with the behaviors and scenes during authoring time and players would then rate the behaviors and scenes according to these constraints. The ratings would then be collected by Second Mind and presented to the authors. During phase III, authors mentioned that they found the numerical ratings to be helpful. It provided them with some feedback on how their behaviors were performing and how the players were perceiving the authored behaviors. Based on that feedback, they were able to modify the behaviors to achieve a better rating. P30 commented on the usefulness of the numerical ratings:

*It (numerical rating) was definitely helpful. It gave me an idea on what was way off, It was useful for sure. You need to have a way to know if you are doing it right or not. The ones that were way off, it was pretty clear what I was missing. The ones that were closer together I guess it would be a little harder on why it was a problem. (P30)*

The numerical rating as P30 mentioned provided a way for authors to know if what they were authoring was perceived well by the players or not. The ratings provided authors with a sense of which action sets would provide a better player experience and they can use these highly rated actions sets to get them a higher score.

The testing scaffolding (in the form of behavior constraints and player feedback) provided by the authoring interface allowed authors to receive the necessary player feedback to modify the behaviors to achieve a better rating than their original behavior sets. The numerical ratings were, however, able to improve the ratings only to some extent. Numerical ratings were not able to provide help on identifying issues with the behaviors like a behavior not getting triggered or directions on how some of the existing behavior sets could be modified. This scaffolding was provided by AI suggestions as discussed next.

### 6.6.2 Debugging scaffolding:

The AI repair provided three kinds of suggestions:

- Trigger suggestion: on how the triggers associated with the existing behaviors could be modified.
- Existing behavior action suggestion: on how existing behaviors could be modified.
- Addition suggestion: on which new behaviors could be added to the existing behavior sets.

During phase III, trigger suggestions were found to be helpful by the authors. These suggestions helped the authors identify problems with the existing behavior triggers. The suggestions provided authors a way to change their existing triggers. P22 mentioned:

*It was very useful, I was able to see what the problems with my triggers were and it helped me find the problems with it and how to address them, that was very good (P22).*

Addition suggestions were also found to be helpful. Addition suggestion helped authors to plug-in the holes in their existing behavior sets. They were suggested interactions (in the form of behaviors) that they had missed during phase I authoring. P23 commented on this during phase III interview:

*Yes, as I mentioned earlier, its natural that some obvious things get ignored sometimes. Detailed suggestions were helpful and situational example helped in analyzing the situation better and provided new things that could be added.(P23)*

The addition and trigger suggestions provided by the AI repair systems helped the authors achieve a better rating than just the numerical rating.

### 6.6.3 Design scaffolding:

In step 2 of phase III experiments, authors were allowed to modify and build upon the suggestions provided by the AI suggestions. Addition suggestions were found to be helpful in providing new ideas on what other new interactions could be added to the existing behavior sets. P10 commented:

*Yes the feedback (AI repair) was helpful in adding new actions to the behaviors. The discovery of new behaviors in the feedback helped me think of some other key behaviors which were missing earlier (P10).*

Authors would take the AI addition suggestion and suggestions on existing behaviors and would modify these suggestions to better suit the kind of player experience they wanted to create. P22 expressed this in his statement:

*The in-depth suggestions would help me by saying to me "have you thought of this stuff", I would then take the idea and apply it to my stuff. I won't be able to use "How would you buy the cloth"(the suggested behavior") as it is, but then I understand the intent and then I will use that and apply it in my own way and may be add some other new behaviors.(P22)*

Authors building upon the suggestions provided by the AI repair system resulted in a higher rating for mixed adapted behavior sets compared to just the AI repair suggestions.

I also conducted a more detailed analysis of the mixed adapted behavior sets. The mixed adapted behavior sets were found to lie in four categories. These categories indicated the percentage distribution of the total changes performed on the original behavior sets between the author and AI. Based on the percentage distribution, following categories were identified:

- **Mixed Adapted I (called MA (0-10 Auth.))**: This category represented player's ratings on mixed adapted behaviors where of all the changes done to the original behavior sets, the changes from the author ranged between 0-10% and the changes performed by the AI ranged from 90-100%.

- **Mixed Adapted II (called MA (10-20 Auth.))**: This category represented mixed adapted behavior sets where of all the changes done to the original behavior sets, the changes from the author ranged between 10-20% and the changes performed by the AI ranged from 80-90%.

- **Mixed Adapted III (called MA (20-30 Auth.))**: This category represented

**Table 9:** The table shows distribution of authors among the different mixed adapted categories.

| **Category** | Number of Authors |
|---|---|
| **Mixed Adapted I** | 6 |
| **Mixed Adapted II** | 7 |
| **Mixed Adapted III** | 7 |
| **Mixed Adapted IV** | 6 |

mixed adapted behavior sets where of all the changes done to the original behavior sets, the changes from the author ranged between 20-30% and the changes performed by the AI ranged from 70-80%.

- **Mixed Adapted IV (called MA (30-40 Auth))**: This category represented mixed adapted behavior sets where of all the changes done to the original behavior sets, the changes from the author ranged between 30-40% and the changes performed by the AI ranged from 60-70%.

The following procedure was used for calculating the percentage distribution between AI and author:

- Each of the AI addition, existing behavior and trigger suggestions accepted by the author as it is (i.e without any changes) were counted as +1 change by the AI. Author additions were likewise counted as +1 change by the author.

- Complete removal of the AI addition, existing behavior and trigger suggestions by the author were counted as +1 change by the author and 0 change by the AI. Partial changes on AI suggestions were counted as +1 change by the AI and +1 change by the author.

- All the change count for the AI and author was summed up and percentage distribution between the author and AI was calculated.

The distribution of authors among these four categories is shown in Table 9. There were no mixed adapted behavior sets that had more than 40% changes from the authors. The results of the ratings distribution are shown in Figure 65. There are three other categories visible on the figure. These categories are the following:

- **Original**: This category represented the original behavior sets of the authors without any adaptation (author or AI).
- **Author adapt.**: This category represented player's ratings on author adapted behavior sets without any AI support.
- **AI adapted**: This category represented player's ratings on AI adapted behavior sets without any author intervention.

As the figure shows, the AI and author adapted behavior sets on their own got lower player ratings when compared to mixed adapted behavior sets where the authors and AI combined together. Furthermore, as the author took a more proactive role by building upon changes suggested by the AI, the ratings improved further. Mixed adapted IV got the highest rating among the four mixed adapted categories followed by Mixed adapted III, II and I respectively. Presenting the decisions of the AI repair system to the authors allowed higher quality behaviors than would have been possible otherwise.

Examples of player interactions with avatars after author adaptation, AI repair and mixed adaptation are presented in Figure 66, 67 and 68 respectively.

## *6.7 Chapter Summary*

In this Chapter, I have presented a four phase evaluation study has been carried out to evaluate the authoring and repair system. In the first phase of the evaluation study, six category of users (UI designers, AI programmers, programmers, Acting background non-programmers, Second Life non-programmers and non-programmers

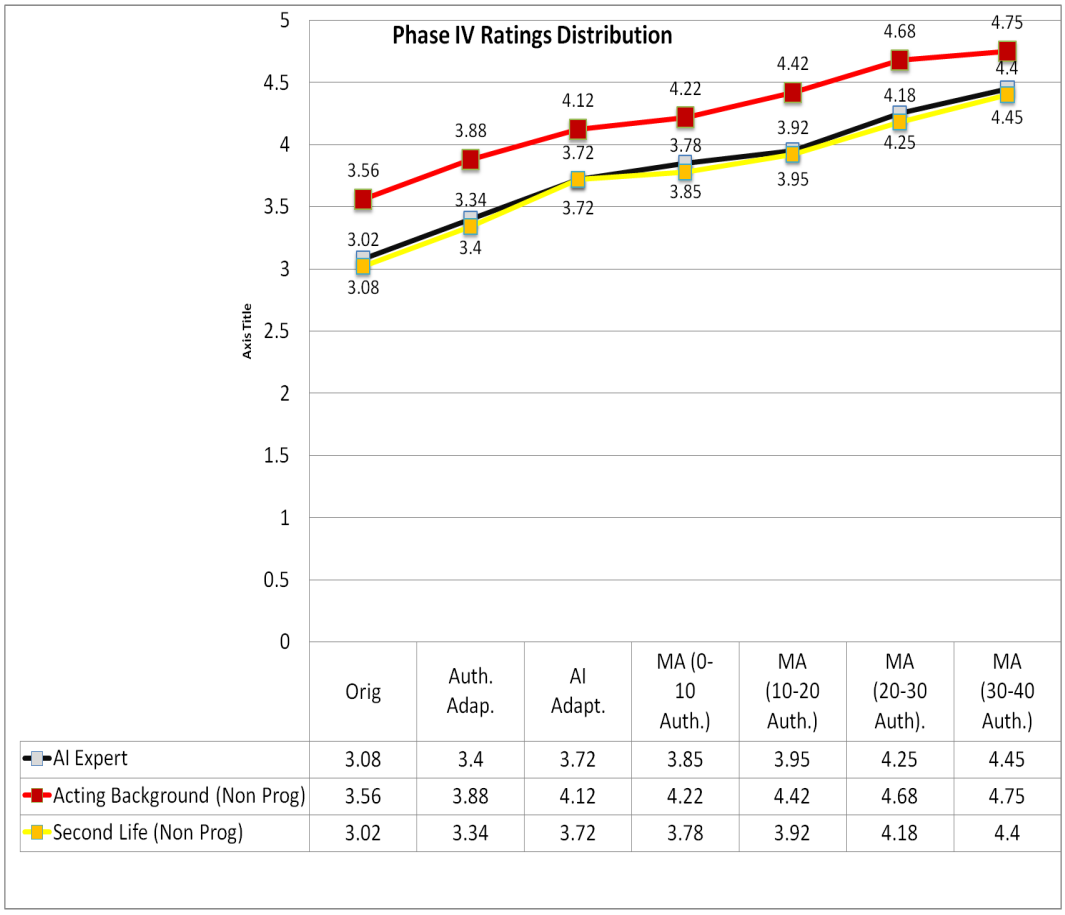



**Figure 65:** The figure shows distribution of player's ratings for different author category behaviors in phase IV. The figure also shows more detailed distribution of mixed adapted ratings based on authors modification percentage compared to the original AI suggestions

with varied backgrounds were invited to author shopkeeper behaviors for a Second Life island using the authoring system. In the second phase, another set of users (I call them players) interacted with the shopkeeper avatars (created as part of the first step) in SL. This provides the necessary feedback for AI repair system to improve the behavior sets. In the third phase, a subset of the same user set who carried out behavior authoring in first phase were invited back. Authors were first presented with text based feedback on how well the behavior sets performed. These text based feedback were simple textual description detailing the performance of the behaviors with respect to associated constraints. The text based feedback is based on feedback provided in second step. The authors modified the behaviors based on the suggestions. In the second step, authors were presented with the suggested changes from the AI repair system. Authors took these AI suggestions as the starting point and either accepted them as it is or adapt the changes further. In the fourth step, another set of players interacted with four sets of behaviors. The first set was the original behaviors created in phase I. The second set of behaviors were the ones that have been modified based on authors themselves by looking at the text based feedback. The third set of behaviors were the ones modified by the AI (without any author intervention) and the fourth set were the mixed adapted behaviors that were a combination of modification by the AI and the authors. Players interacted with the four sets of behaviors in Second Life and provide feedback on their interaction. The ratings for all the four sets of behaviors provided a way of measuring AI repair approach effectiveness in appropriately adapting the behavior sets and achieving a better player experience.

The results indicated the following.

- **Novice user understandable AI behavior authoring environment:** The authoring system provided a novice user understandable AI behavior authoring environment for authors to construct the behavior sets through,
  - *Novice user understandable AI vocabulary:*

- *Understandable interaction design approach:*
- *Executable framework for behavior evaluation:*

- **Advanced AI scaffolding:** The repair system provided advanced AI scaffolding for design, debugging and testing of authored behaviors to help the authors improve their behavior sets.
  - *Design scaffolding:* in the form of new ideas on adding behaviors and thereby creating a richer player interaction.
  - *Debugging scaffolding:* in the form of helping the authors to debug the authored behavior sets and identify problems with them.
  - *Testing scaffolding:* in the form of providing a framework for authors to assign constraints on the authored behavior sets and gather player feedback on the constructed behaviors.

| | |
|---|---|
| **A(1):** | ~~hey good morning.~~<br>Gesture ("smile")<br><br>Hello and welcome to our store. We have a great collection of furniture items. I am sure you are going to enjoy visiting us today.<br>Gesture ("wave")<br><br>Gesture ("peace") |
| **U(1):** | User clicks on button "say hello". |
| **U(2):** | User clicks on button "inquire". |
| **A(2):** | What are you looking for in the shop? There are special offers going on in several items today! If you buy any item which is pick in color would have extra 50% discount on it..<br>Gesture ("surprise")<br><br>Gesture ("point me")<br><br>Gesture ("jump for joy")<br>*(...user is presented with choices, "ask for sale", "sofa sets", "chairs")* |
| **U(3):** | User clicks on button "ask for sale". |
| **A(3):** | We have an extra 50% discount today. We also have additional sale which is buy one and get one free. We have some great offers going on and people have liked these offers a lot. I also wanted to tell you that these offers are just for today.<br>Gesture ("laugh short")<br><br>Gesture ("Surprised") |
| **A(4):** | Let me know if you need any assistance! I am around and feel free to ask anything. Just for your information, the closest price checker is at the front side of the shop.<br>*(...user is presented with choices, "two-seater sofa", "three-seater sofa", "combination-sofa")* |
| **U(4):** | User clicks on button "two-seater sofa". |
| **A(5):** | Oh these are great items which you've selected to buy! You are almost getting these at peanut prices! I'll do the billing!.<br><br>~~Gesture ("tantrum")~~<br><br>Gesture ("toothsmile")<br>*(...user is presented with choices, "bye", "happy with the purchase")* |
| **U(5):** | User clicks on button "bye". |
| **A(6):** | Thank you and have a good day. Please visit us again sometime. |

**Figure 66:** Example of modifications from the phase III study by a Second Life non-programmer. Blue shows modification carried out by the author.

| | |
|---|---|
| **A(1):** | Hey there. Great day isn't it?.<br>Gesture ("laugh short")<br>Gesture ("point you"), Gesture ("wave")<br>Welcome to my store. How could I help you? We have a nice collection of clothes that I am sure you are going to enjoy.<br>Gesture ("smile")<br>*(...user is presented with choices, "ask for help", "ask about clothes", "chit chat", "talk about marketplace", "talk about weather")*<br>*(...silence for some time)* |
| **A(2):** | Let me know if you need any assistance! I am around and feel free to ask anything. |
| **U(1):** | User clicks on button "talk about weather". |
| **A(3):** | We have been having a great weather these days. I am sure you are going to enjoy it here.<br><br>Gesture ("Smile"), Gesture ("Wink")<br>*(...silence for some time)* |
| **A(3):** | We have a nice collection that has come in. What would you like to have today?.<br><br>Gesture ("point towards the player")<br>*(...user is presented with choices, "my options?, "deals", "sale items")* |
| **U(2):** | User clicks on the button "deals". |
| **A(4):** | Well we have a great deal on these clothing. You can get one shirt free for any two shirts you buy. All jeans are 50% off<br><br>Gesture("Shrug"), Gesture("jump for joy") |
| **U(2):** | User goes close to the second room. |
| **A(5):** | There are some new designer jeans and some nice summer collection that has just arrived.<br>*(...user is presented with choices, "designer jeans", "summer collection", "casual shirts")* |
| **U(3):** | User clicks on button "designer jeans". |
| **A(6):** | That sounds excellent. This is a great choice. Customers have really enjoyed them.<br>Gesture ("Point You")<br>I bought one of these myself. |
| **U(5):** | User clicks on button "purchase item". |
| **A(8):** | This is going to be wonderful for you.<br><br>I am sure you are going to enjoy the purchase. |
| **U(6):** | User clicks on button "compliment user". |
| **A(9):** | Gesture ("Surprise")<br><br>Oh Thanks. This is great. I am glad you liked it.<br><br>Would you like anything else today? |
| **U(7):** | User clicks on button "goodbye". |
| **A(11):** | Gesture ("wave")<br>Well, thanks for visiting us. Hope you see you again soon!<br>Gesture ("toothsmile") |

**Figure 67:** Example of modifications from the phase III study. User interacts with the avatar created by an author who is a Second life non-programmer. Red shows the modification carried out by AI.

| | |
|---|---|
| **A(1):** | Hello there, welcome to carl's furniture shope. how can I help you?<br>Gesture ("smile")<br><br>Gesture("wave")<br>*(...user is presented with choices, "need a couch", "need chair", "need bed" "need teek chair", "need directions")* |
| **U(1):** | User clicks on button "need a couch". |
| **A(2):** | Gesture ("point you")<br>what do you think of awesome leather one. We have a special price today.<br>Gesture ("toothsmile")<br>*(...user is presented with choices, "ask for sale", "sofa sets", "chairs")* |
| **U(3):** | User clicks on button "ask for sale". |
| **A(3):** | I have exactly the couch for you.<br><br>Gesture ("point me") |
| **U(4):** | User clicks on "purchase item". |
| **A(4):** | your total is $300.<br>come with me<br>Absolutely, we have excellent financing in this shop. I can offer you a 12 month warranty<br>You can see the conditions in this booklet<br>we would need some down payment as well<br>Gesture ("point you")<br>how would you like to pay?<br>*(...user is presented with choices, "check-out(cash)", "check-out(credit-card)"* |
| **U(5):** | User clicks on button "check-out(cash)". |
| **A(5):** | Cash is excellent. We will ship it in two days.<br><br>Gesture("frown")<br><br>Gesture ("tantrum")<br>*(...user is presented with choices, "offer credit card signing", "inquire group-membership"))* |
| **U(6):** | User clicks on button "inquire group-membership". |
| **A(6):** | Would you like to sign up for our group membership?<br><br>Gesture("smile"), Gesture("tantrum") |
| **U(7):** | User clicks on button "accept group-membership". |
| **A(7):** | Thank you. we will keep you posted on any future updates.<br><br>Gesture("Surprise") |
| **U(8):** | User clicks on button "Goodbye". |
| **A(8):** | Good bye. Thanks for visiting us here<br>Gesture("Salute") |

**Figure 68:** Example of modifications from the phase III study. User interacts with the avatar created by an author who is an AI programmer. Blue shows the modification carried out by the author himself. Red shows the modification by the AI

# CHAPTER VII

# CONCLUSION AND FUTURE WORK

## *7.1 Overview*

Creating high quality AI behaviors for interactive characters is a hard problems. There are two reasons for it.

- First, creating AI behavior sets require specialized skills in development and coding. The nature of the process restricts it to certain individuals who have expertise in this area. There is little understanding of how the behavior authoring process can be simplified with easy-to-use authoring environments so that novice users ((without programming and design experience) can carry out the creative task.

- Second, constructed AI behavior sets have problems which cause a break in player experience when they repeatedly fail. It is harder for novice users who do not have the debugging and design experience to identify and correct problems with the authored behavior sets and improve them.

In this thesis, I explored the issues related to the problems highlighted and proposed a solution to them within an application domain, named Second Mind(SM). In SM novice users who do not have expertise in computer programming design behaviors for intelligent virtual characters. These behaviors are essentially sequence of basic actions (like talk, gestures etc) that can be conducted in the game world and can be triggered in response to the game world state. An AI approach in the authoring environment supports the authors in identifying and repairing problems with their authored behavior sets and helps them improve their quality through a better

experience for players who interact with the authored behavior sets.

To evaluate the solution I conducted a four part evaluation study with human subjects. The results indicate that the behavior authoring and repair solution is able to provide authors with a) programming scaffolding through easy to use authoring environment, understandable vocabulary and integration with a virtual world to construct the behaviors, b) testing scaffolding to receive feedback from the players, c) debugging scaffolding to help the authors identify problems with the authored behavior sets and d) design scaffolding by suggesting them new behaviors they could add to their existing behavior sets. The design, debugging and testing scaffolding improves the quality of the authored behavior sets by achieving a better experience for players who interact with the authored behavior sets.

## *7.2 Recap: Research Questions, Answers and Claims*

One of the key challenges in making the AI behavior construction process simpler is to select interaction design approach, languages, and tools that enhance the authoring environment's understandability. This raises the following key research question that need to addressed:

- **Research Question 1: What is a design solution that enables novice users to easily create AI behaviors for NPCs performing a service in a virtual world?**:

Users should be able to infer how the programming system works by referring to their existing knowledge and expectations about how the modeled system works. My answer to this question is the following:

A design solution with behavior timeline based interaction design approach for behavior construction supported by an understandable vocabulary and reduced feature representation formalism enables novice user to create AI behaviors in an easy and understandable way for NPCs performing a service in a virtual world.

The second research question is the following:

- **Research Question 2: How can a system automatically identify failures in human authored AI behaviors for NPCs performing a service in a virtual world?**

My answer to this question is the following:

Comparison of Successful and Unsuccessful execution traces can be used to identify failures in human-authored AI behaviors for NPCs performing a service in a virtual world.

The third research question is the following:

- **Research Question 3: How can a system repair failures in human authored AI behaviors to improve player experience?**

My answer to this question is the following:

Execution traces where the identified failures do not exist allow successful repair of AI behaviors for NPCs performing a service in a virtual world. The success of repair is measured in terms of improvement in the player experience ratings of the repaired AI behaviors compared to novice users original unrepaired and self-repaired AI behaviors.

### 7.2.1 Thesis Claims

One the basis of work conducted in this thesis, I make the following claims:

I. **A design solution with behavior timeline based interaction design approach supported by an understandable vocabulary and reduced**

**feature representation formalism enables novice users to author AI behaviors in an easy and understandable manner. The scope of AI behavior creation is confined to creation of action sequences for NPCs performing a service in a virtual world that can be activated based on author defined activation triggers.**

II. **An introspective reasoning approach based on comparison of successful and unsuccessful execution traces can be used as a means to successfully identify breaks in player experience and modify the failures to improve the experience of the player interacting with NPCs performing a service in a virtual world.**

The scope of construction and repair of the AI behaviors is restricted to action sequences for non-playing characters performing a service in a virtual world. These services range from a shopkeeper to a museum host.

The claims leads to the following thesis statement:

**An introspective reasoning approach for repairing behavior failures supported by a behavior timeline based behavior construction process, understandable vocabulary and reduced feature representation formalism enables novice users to achieve behavior quality comparable to AI experts. The scope of AI behavior creation and repair is confined to creation of action sequences for NPCs performing a service in a virtual world that can be activated based on author defined activation triggers.**

## *7.3 Solution Summary*

The authoring solution (named Second Mind) consists of three parts. First, behavior authoring process is represented as designing agents mind in a graphical environment. The user is presented with basic building blocks that can be combined together on a timeline to construct behavioral entities. Timelines are employed by

most popular professional video editing packages and builds on existing interaction design approaches used by novice users. Timelines indicate a left-to-right sequence, where each event occurs in chronological order. The behavior construction process represented as a sequence of actions on a timeline in a mind design editor makes it easier for novice users to conceptualize the process. Second, through an iterative development approach, a representational formalism and vocabulary for authoring process is created. The vocabulary is made intuitive and understandable for novice users. The complexity and feature set of the representation are reduced but it is still sufficient enough to represent and execute the behaviors. Third, a behavior execution framework is provided that executes the authored behavior in a concrete virtual world environment named Second Life (SL), thus allowing novice designer to observe the simulation of resulting behaviors showing the interaction with other characters.

The work is focused on creating island hosts in SL. One of the problems currently in SL is the empty islands when you visit these virtual spaces. The authoring system is targeted towards creating island hosts in SL. These island hosts are created for various different types of virtual spaces in SL and perform different kind of roles. These roles range from shopkeepers selling goods like furniture's, DVD's and clothes or a receptionist for a university department.

Once the behavior sets are authored, they are repaired using an approach consisting of three parts. First during authoring, behaviors and overall interaction are associated with constraints that provide a way to measure the success of the behavior sets and the overall interaction. Second, players (who interact with the authored avatars) provide feedback at the end of their interaction by answering questions on executed behaviors performance and overall interaction. These questions ask for feedback on the performance of the behaviors in relation to the associated behavioral constraints and overall interactions(defined in the first step). These numerical feedbacks help measure the success of the behaviors and overall interaction in relation

to the associated behavioral constraints and overall interaction constraints. Third, the repair of behaviors is carried out through comparison of successful and unsuccessful execution traces. Successful execution traces have a high overall interaction rating whereas unsuccessful traces have a low overall interaction rating provided by the player. The comparison of successful and unsuccessful traces provides a way to identify failures in the unsuccessful behavior sets. Once the problems are identified, their modification involves looking at execution traces (where these failures do not exist) to suggest modifications to the author.

## *7.4 Evaluation Approach*

I conducted a four phase evaluation study. In the first phase of the evaluation study, six category of users (UI designers, AI programmers, programmers, Acting background non-programmers, Second Life non-programmers and non-programmers with varied backgrounds are invited to author shopkeeper behaviors for a Second Life island using the authoring system. In the second phase, another set of users (we call them players) interact with the shopkeeper avatars (created as part of the first step) in SL. This provides the necessary feedback for AI repair system to improve the behavior sets. In the third phase, a subset of the same user set who carried out behavior authoring in first phase are invited back. Authors are first presented with text based feedback on how well the behavior sets performed. These text based feedback are simple textual description detailing the performance of the behaviors with respect to associated constraints. The text based feedback is based on feedback provided in second step. The authors modify the behaviors based on the suggestions. In the second step, authors are presented with the suggested changes from the AI repair system. Authors takes these AI suggestions as the starting point and either accepted them as it is or adapt the changes further. In the fourth step, another set of players interact with four sets of behaviors. The first set is the original behaviors created

in phase I. The second set of behaviors that have been modified based on authors themselves by looking at the text based feedback. The third set of behaviors is the one modified by the AI (without any author intervention) and the fourth set is the mixed adapted behaviors that are a combination of modification by the AI and the authors. Players interact with the four sets of behaviors in Second Life and provide feedback on their interaction. The ratings for all the four sets of behaviors provide a way of measuring AI repair approach effectiveness in appropriately repairing the behavior sets and achieving a better player experience.

## *7.5 Evaluation Results*

### 7.5.1 Phase I results

Phase I analysis provided the following results:

- **Programming scaffolding for non-programmers:** Non-programmers were able to author behaviors using the scaffolding provided by the authoring system through a) understandable terminology and representation vocabulary, b) easy to use authoring environment and c) virtual world connection that helped them identify problems with the authored behaviors.
  - **Understandable terminologies and representation vocabulary:** Authors were able to understand the terminology and representational vocabulary (like scenes, behaviors triggers, actions etc). Based on the quantitative ratings and qualitative feedback, non-programmers had an equal understanding of the terminology on par with AI behavior programmers and programmers.
  - **Ease to use interaction design approach:** Non programmers found that they did not need any programming experience to use the system and provided numerical ratings (on ease of use) which were statistically similar in rating to AI behavior programming experts. Authors felt that using the

timeline editor, the system allowed them to be very creative where they could go in the authoring interface and create in a small amount of time any kind of world that they had imagined.

  - **Virtual world connection with the authoring interface helped authors identify problems with behaviors:** The connection to a virtual world (Second Life) was very helpful for authors in identifying problems with the created behaviors. Authors created the behaviors in the authoring interface and then went into SL, interacted with the authored behaviors and identified any issues with the behaviors.

- **Authoring interface limitations:** Authors requested the following features to be added in next version of the system a) video tutorials, b) advanced programming features and c) Animation preview option and new animations.

  - **Video tutorials:** Authors felt that it would have been very helpful to have more video tutorials that they can just view instead of a tutorial file. The video tutorial would have made the process more easier for them.

  - **Advanced programming features:** Authors esp. programmers wanted to have other advanced features like random behaviors or behaviors that could be selected based on probabilities. They also wanted to have features where they could go in and view the internals of the behaviors (like looking at the generated code) to modify it.

  - **Animation preview and new animations:** Authors found that not having an animation preview button was a limitation of the system. An animation preview button would have allowed authors to watch the avatar's animation right in the authoring interface itself (currently they had to go in second life and view the results of the animation they had selected). Authors also wanted the ability to create new animations that would have

allowed them to create newer kind of behaviors.

### 7.5.2 Phase II results

Results from phase II indicated the following:

I. **Acting background authors got the highest rating.**

II. **Second life non-programmers, non-programmers with misc. background and AI behavior programmers got statistically similar ratings and had the second highest ratings.**

III. **UI designers and programmers got the lowest rating.**

Interview data from phase I provided some answers to these questions. Player interaction from phase II and analysis of the authored behaviors also provided a better understanding of the reasons behind these differences. The analysis indicated the following:

- **Programming scaffolding helped the non-programmers:** The authoring interface provided non-programmers the necessary support through easy to use authoring interface, understandable vocabulary and concrete virtual world feedback on the authored behaviors allowing them to conduct the authoring process and create behavior sets that were either on par (in case of non-programmers with misc. background and Second Life non-programmers) or better (in case of acting background non-programmers) than AI behavior programmer created behavior sets.

- **Authoring interface supported author's creativity:** Authoring interface allowed the authors to detail the behaviors, express their creativity and use the design principles in authoring the behaviors.

- **Programmers and UI designers focussed on the limitations, unable to find work-around to create complete player experiences:** Programmers found that the authoring interface didn't provide them with advanced programming features to create behaviors. They were not able to work around these limitations (that other authors were able to) and use the existing authoring interface to create complete player interactions. UI designers were similarly unable to work-around the usability limitations.

### 7.5.3 Phase III results

Phase III analysis provided us with the following results:

- **Design scaffolding through addition suggestions**: AI addition suggestions provided the necessary design scaffolding for authors.
  - The addition suggestions helped complete the missing interactions within a scene. The addition suggestions generated by the AI repair system also helped the authors by giving them new ideas on additional behaviors that could be added to the scenes.
- **Debugging Scaffolding through trigger suggestions:** AI trigger suggestions helped the authors in identifying problems with the behaviors and correct them.
  - The trigger suggestions generated by the AI repair system were very helpful (esp. for non-programmers) in finding problems with the behaviors.
- **Numerical ratings were helpful but didn't tell "how" to modify the behaviors:** The numerical ratings provided in the first step were found to be helpful to some extent by the authors. The numerical rating provided authors with an idea of player's perception of the behaviors and the authored scene. Although numerical ratings were helpful, they did not however, provide authors

with an idea on what was missing from the behaviors and how they could modify the behaviors to achieve better ratings.

- **Limited justifications on AI suggestions**: AI suggestions were found to be limited in terms of providing the justifications on why an AI suggestion was being made.
  - The suggestions produced by AI repair system were found to be limited in providing detailed reasoning on "WHY" an AI repair suggestion was made. This missing reasoning explanations would have been helpful in achieving more buy-in from the authors.

### 7.5.4 Phase IV results

Phase IV results indicated the following:

**Mixed adapted behavior sets across the different author categories got the highest player rating followed by AI adapted, author adapted and original behavior sets.**

Interview data from phase III provided some answers to these questions. Player interaction from phase IV and analysis of the authored behaviors also provided a better understanding of the reasons behind these differences. The analysis indicated the following:

- **Testing scaffolding** in the form of numerical feedback from players helped authors understand the behaviors that were not working well and helped them adapt the behaviors.
- **Debugging scaffolding** in the form of feedback on triggers and existing behaviors led to AI adapted behaviors receiving a higher rating than author adapted behavior sets

- **Design scaffolding** in the form of addition suggestions that provided authors with new ideas on things that could be added to the behavior sets in order to create a more richer interaction.

The development and evaluation of Second Mind authoring and repair system also revealed certain limitations and future directions. I discuss these next.

## *7.6 Limitations and Future Directions*

There is much work left to do in Second Mind, some of which is research, some of which is development. There are also numerous open questions that form possible avenues of future directions for this research. In this section, I will provide some of the future directions that Second Mind could take to make the behavior authoring easier.

### 7.6.1 Authoring Support through demonstrating behaviors

In phase III, the repair system provided authors with addition suggestions on their existing behavior sets. Example of an addition suggestion was a payment method behavior for allowing players to pay for the bought item. Accepting payment from the customers can be considered an important part of a shopkeeper scenario, however some of the authors missed the behavior during the authoring process. During phase III interviews, authors mentioned that if they were actually performing the scenario (like acting as a shopkeeper), then they would have probably not missed these behaviors. P30 mentioned:

*If I was actually doing it, it would have occurred to me to add the payment behavior but since I was directing it in a sense (in the authoring interface) it didnt occur to me. So I am here sitting acting as a director and computer is the actor. I am here telling it what to do and it is responding to what you are asking it to do and if its doing something wrong then you as a director are doing something wrong whereas*

*as an actor you are in the moment, you are actually performing there, and then its hard to miss. (P30)*

Allowing authors to perform the behaviors apart from using a behavior editor to create them would perhaps address the missing behavior issue and make the process more easier for the authors. The authors could perform and act out a scene in the virtual world and the system could learn from these performances and create the first version of the behaviors which the authors can then edit in the authoring interface. Let's look at a prospective approach that could support such a possibility.

**Learning from demonstration approach:** Human learning is often accelerated by observing a task being performed or attempted by someone else. In fact, infants spent a lot of their time repeating the observed behaviors [92]. These capabilities of the human brain are also evident in computer games where players go through a process of training and imitating experienced players. These results have inspired researchers in artificial intelligence to study learning from imitation techniques. By observing an expert's actions, new behaviors can quickly be learnt that are likely to be useful; because they are already being used by the expert successfully. I would like to incorporate an approach in the authoring interface that utilizes this ability to extract behavioral knowledge for computer games from expert demonstrations.

A prospective architecture for such a system where the authors demonstrate the behavior to be learnt (by controlling their avatar manually) instead of having to code the behavior using a programming language and the system learns from that demonstration is shown in Figure 69). The architecture would consist of four steps:

- *Demonstration*: The player plays the game, demonstrating the particular behavior he wants the system to learn. This process results in a trace, i.e. a log file that contains each action that the expert executed, together with their respective game state and time stamps.

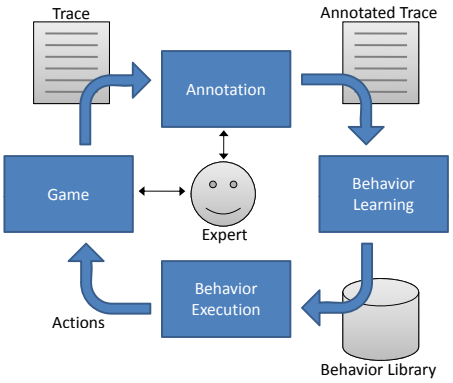



**Figure 69:** The figure shows Learning from Demonstration Architecture, involving 4 steps: demonstration, annotation, behavior learning and finally behavior execution.

- *Annotation*: The player annotates the trace specifying which goals (selected from a predefined set of goals) he was attempting with each action. Annotation could be performed using an easy to use GUI.

- *Behavior Learning*: The annotated trace is handed to a behavior learning module, which can automatically extract procedural behaviors from the annotated trace, and store them in a behavior base.

- *Behavior Execution*: Once the behavior base has been populated, the learnt behaviors can be executed in the game using the behavior execution engine.

I have experimented with learning from demonstration approach in other domains [74] with successful results.

### 7.6.2 Supporting UI designers and Programmers

Phase I results indicated that UI designers were unable to create 10.13% of the behaviors that they had envisioned for the shopkeeper avatar. UI designers wanted to create behaviors where the players would be allowed to interact with the avatar through richer sets of inputs (instead of the current button-based interaction). UI designers also wanted a way to change the current button-based GUI so that they could allow the players to express their emotions (through emoticons for example)

which the avatar could then react to. They also wanted a way to modify the layout of the button based player UI that the players used to interact with the avatar. The current authoring interface was unable to provide these features.

From the above comments, it can be seen that UI designers wanted to enable players to have better interaction possibilities with the avatar through a) Better UI for interacting with the avatar and b) richer input set during their interaction with the avatar. Let's look at how some of these features could be supported in future versions of Second Mind.

#### *7.6.2.1 Richer player input: Templatized NLU Input*

The player input is currently restricted to clicking on buttons to interact with the avatars. One way to allow richer input from the players would be to allow them to interact in natural language. In order to do that, authors could create pattern matching templates in the Second Mind authoring interface for understanding player's natural language input, an approach similar to what has been used to create chat-bots like ALICE[1]. This would allow players to express themselves in natural language instead of a button-based interaction. In order to limit the kind of templates that the author needs to create, an inverse parsing based approach[2] could be provided where the player is suggested acceptable language formations as he types in input to interact with the avatar.

#### *7.6.2.2 Support for player UI modification*

Currently the authors are not provided the ability to modify the layout of the user interface that player uses to interact with the avatar. The current UI for player interaction is shown in Figure 70. To modify the player UI, authors could be provided an editor that allows them to create new UI for player interaction.

[1] http://alice.pandorabots.com/

[2] http://www.erasmatazz.com/page78/page31/page303/page306/InverseParser.html

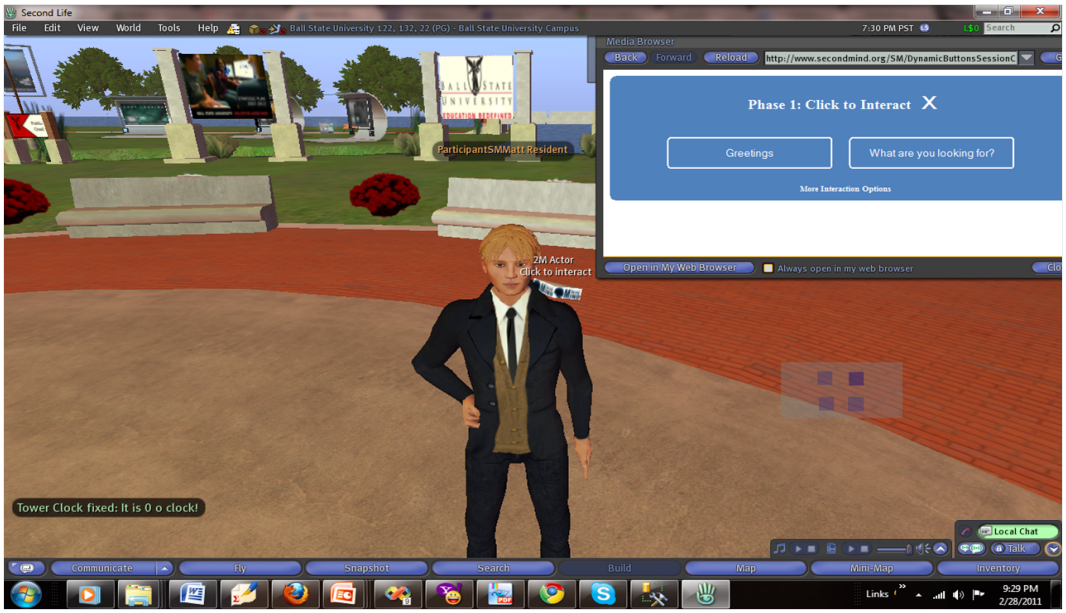



**Figure 70:** The figure shown a screenshot of the player interaction in Second Life.

#### *7.6.2.3 Supporting programmers through advanced programming features*

User study provided data on certain advance features that were found missing from the authoring interface (that programmers would like to have). The authoring interface didn't have a code viewer where authors could go in and view the internals of the behaviors to modify it. One way to address this issue would be to have an intermediate XML representation that is generated based on the database representation of the behaviors and then the XML representation could be presented in a code editor so that the authors could modify the behaviors directly by modifying the XML representation. Authoring interface also missed features like the possibility of creating behaviors that could be carried out by the avatar in random order or behaviors that could be selected based on probabilities. Features like creating random and probabilistic behaviors could be provided by appropriate modifications in the behavior execution system and then presenting this feature in the authoring interface to allow authors to create such behaviors.

### 7.6.3 Limitations of behavior authoring using the current authoring interface

As discussed in Chapter 4, the current authoring interface provides novice authors the ability to create AI behaviors that are a sequence of actions that can be carried out in the virtual world. These behaviors can be triggered by the interacting player or the events happening in the virtual world. As we found in Chapter 6, the AI behavior programmer's were able to find workaround for 3.8% of the behaviors that they wanted to create and had initially sketched on paper. The evaluation study also did not bring up any limitations on the trigger sets. Even though the evaluation study did not highlight any specific behavior creation limitations, the authoring interface has certain limitations that need to be worked upon in order to scale it up for larger and a variety of different domains. Some of the limitations and way of addressing them have already been mentioned before in the context of supporting UI designers and programmers. I will discuss some of these other limitations and improvements in the authoring interface that could overcome the limitations.

#### *7.6.3.1 Hierarchical behaviors*

Chapter 3 highlighted the problems of presenting the authoring task as a hierarchical goal subgoal structure. As discussed in Chapter 4, in order to address the issue, we used the notion of timelines that presented the authoring task as creating a left to right sequence of actions. As a result, in the current authoring interface the authors are unable to create hierarchical behaviors. The limitation could be addressed by supporting addition of already authored behaviors on the existing timeline (shown in Figure 71) when a new behavior is being created. This would allow authors to link behaviors in a hierarchical structure while still using the existing timeline based approach in the authoring interface.

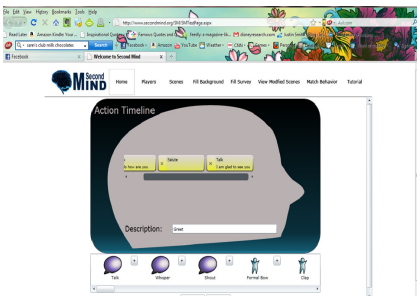



**Figure 71:** The figure shows the timeline based behavior authoring interface.

#### *7.6.3.2 Trigger set limitations*

As we discussed in Chapter 6, the evaluation study didn't highlight any trigger limitations. Perhaps the kind of scenes (shopkeeper) the authors were asked to create didn't result in limitations on the triggers. For more advanced authoring scenarios, an editor which allows authors to construct complex triggers would need to be provided. These editor would allow creation of newer kind of triggers which are created based on defining a variety of mathematical operators on the basic world state information.

### 7.6.4 Second life overhead and connection with browser based virtual worlds

One of the current limitations in the current web-based authoring system is that it requires a separate virtual world client (Second Life) in order to use Second Mind and see the behaviors and interactions created through Second Mind. I decided to use Second Life as the client for three reasons. First, there is a large user base that is present inside Second Life. Second, it allows for easy connection to external controllers through an API. Third, it provides a rich set of basic actions (in the form of emotions and physical actions) [2] that can be executed by the avatar. I also explored the possibility of connecting Second Mind to browser based virtual world. I first explored Metaplace[3] which was intended as a web-based platform for user generated virtual

[3]http://en.wikipedia.org/wiki/Metaplace

worlds. The overall vision behind integrating these type of virtual worlds was that it would allow authors to select/create a browser based virtual world embedded in Second Mind itself without a separate download and a separate application to run in order to execute their behaviors. I also explored Lively[4] which was network of avatars and virtual rooms created and decorated by its users. These virtual rooms could also be embedded like a youtube video on external websites. This would have allowed the possibility of embedding a virtual room like a clothes shop right with the shopkeeper scene in the Second Mind website. Unfortunately both of these virtual worlds did not progress and were discontinued. A future direction for Second Mind to gain traction would be to connect it to an open browser-based virtual world. This would allow Second Mind to leverage the social networking landscape where people could easily share their creations on other social networks like facebook and invite their friends to interact with their creations.

### 7.6.5 Missing detailed reasoning for repair system produced suggestions:

The suggestions produced by AI repair system were found to be limited in providing detailed reasoning on "WHY" an AI repair suggestion was made. Currently the reasoning provided to the authors was the following (for trigger suggestions, for example):

> *In interactions with the user, it was observed that the behavior is sometimes not performed/triggered by the avatar due to the current triggers that have been defined. Other similar behaviors that have received higher ratings and have performed seemed to have the modified trigger that is suggested.*

One way to deal with the issue could be to provide a visualization of the AI decision making process in selecting a suggestion. For example, apart from the suggestions, authors could be presented with a graphical representation of the differences found

[4]http://www.lively.com/

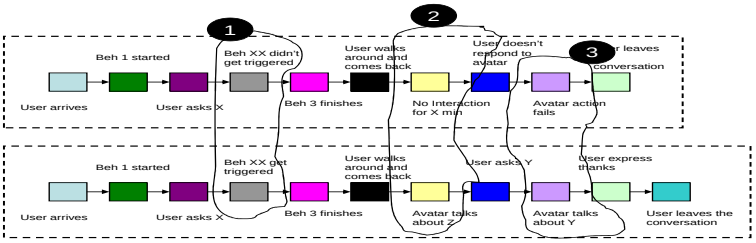



**Figure 72:** The figure shows differences marked in the execution traces

between the successful and unsuccessful traces something along the lines of what is displayed in Figure 72. Examples of execution traces which are used to generate suggestions could also be displayed to help the authors understand the process.

Another approach to explain the internal processing of the repair system could be to use existing approaches that have been used to explain the internal processing of an AI system and have been researched quite extensively [61, 11]. MYCIN, for example, was a medical diagnosis system and used a complex set of rules to identify illnesses and suggest treatments based on patient statistics and test results [97]. In these systems, the idea of providing system the ability to generate explanations was incorporated from the very beginning. The reasoning was that in order for the experts to accept system's suggestions(diagnosis), it would be important for the system to justify its diagnosis.

In order for the repair system in Second Mind to provide an explanation using existing approaches, it would be necessary for it to record the chain of decision making that led to a suggestion. It would also require natural language generation templates [43] to present this chain of reasoning in a human understandable form. The recording of this extended reasoning as well as natural language templates to present this reasoning would allow the repair system to explain its reasoning mechanism in more

detail than the current reasoning that is provided by it.

### 7.6.6 Automatic detection of player's feedback

AI repair system currently requires players to provide input at the end of their interaction on the behavioral constraints and overall interaction constraints for its functioning. Methods through which player feedback could be detected using some automatic means would be helpful and asking players to provide feedback at the end of their interaction. Sensing users physiological activity during the authoring session could be one such automatic approach for capturing player feedback could be avoided. Sensing users physiological activity in human-computer interaction has become quite popular due to availability of affordable sensing technology and recent research advances in interpretation of user's physical states [87]. Measuring player's physiological states during his interaction with the Second Mind authored avatar then using it for repair of authored behaviors would avoid asking player's to provide feedback at the end of their interactions.

### 7.6.7 Supporting iterative revision

As discussed in section 5.3.3.3, in order to repair a particular selected difference, ranking scheme to pick a suggestion from the suggestion list was used. The highest ranking suggestion is selected for repairing the difference and is eventually presented to the author in the authoring interface. Currently when the repair system generates a suggestion, the author can accept the suggestion as it is or can go in and modify the suggestion. Another version of the repair system can be envisioned where the system could take an active part in brainstorming ideas as the author is changing the suggestions. If the author rejects the suggestion that is presented to him in the authoring interface, the system could then present another suggestion from the generated suggestion list and this process could go on resulting in an iterative revision cycle with the author.

### 7.6.8 Supporting suggestions during authoring time

In the current approach, there is currently a disconnect between the authoring session and the repair system providing suggestions to the author. The reason for this disconnect is that we have to wait for the player feedback after the authoring session for the repair system to generate its suggestion list. As a result, the repair suggestions are not generated during the authoring session which would have been quite beneficial for the authors. One way to deal with the issue would be that once a significant number of scenes and behaviors have been authored for a particular context (like different kinds of shopkeepers), the difference and suggestion list based on successful and unsuccessful trace comparison for this particular context is stored. These could then be used to help the author when he is creating a similar scenario. During authoring time, if the system detects that the current author is using a trigger that has not worked for a similar behavior(s), the author could be suggested a modified trigger. Similarly other suggestions could be made during authoring time based on stored difference and suggestion list.

### 7.6.9 Animation preview in the authoring interface

In the current authoring interface, authors were unable to view the animation they selected for creating a behavior. Another feature in the authoring interface like an animation preview button would allow authors to watch the avatar's animation right in the authoring interface itself (currently they had to go in second life and view the results of the animation they had selected). Animation preview feature could be provided by recording videos of the animations in Second Life and allowing authors to view them when they would like to preview the animations in the authoring interface.

### 7.6.10 Video tutorial help:

User study results indicated that it would be helpful to have more video tutorials that can be just viewed instead of a tutorial file (that is currently used). The video

tutorial would have made the process easier for authors. This support could be easily provided by creating some videos which details on the various steps listed in Chapter 4.

## *7.7 Conclusion*

To restate my thesis: **An introspective reasoning approach for repairing behavior failures supported by a behavior timeline based behavior construction process, understandable vocabulary and reduced feature representation formalism enables novice users to achieve behavior quality comparable to AI experts. The scope of AI behavior creation and repair is confined to creation of action sequences for NPCs performing a service in a virtual world that can be activated based on author defined activation triggers.** I have presented design and implementation of an authoring and repair approach and conducted a four part evaluation study to support our my thesis. To sum, the contribution made by this work are the following:

- **Novice user understandable authoring environment** to help them author AI behaviors in an easy and understandable manner for NPCs performing a service in a virtual world,
  - *Novice user understandable AI vocabulary:* for representing the authoring task.
  - *Understandable interaction design approach:* that helps the authors easily create behaviors.
  - *Executable framework for behavior evaluation:* for evaluating constructed AI behaviors.
- **Design, debugging and testing scaffolding** to help novice users achieve higher quality AI behaviors compared to their original unmodified behaviors,

  - *Design scaffolding:* that provided authors with new ideas on adding behaviors and thereby creating a richer player interaction.
  - *Debugging scaffolding:* that helps the authors to debug the authored behavior sets and identify problems with them.
  - *Testing scaffolding:* a framework for authors to assign constraints on the authored behavior sets and gather player feedback on the constructed behaviors.

- **A novel introspective reasoning approach** for successfully detecting and repairing failures in AI behaviors for NPCs performing a service in a virtual world,
  - *Detecting need for repair:* through declarative behavior constraints assigned by the authors and player feedback that provides feedback on the constraints after the interaction with the avatar.
  - *Failure Identification:* A novel approach for identifying the failures through comparison of successful and unsuccessful execution traces.
  - *Repair Methodology:* A novel approach for repairing the failures using execution traces where these failures do not exist.

# APPENDIX A

# PAPER PROTOTYPE STUDY

## *A.1 List of Basic Actions and Percepts*

Figure 73 shows the list of basic actions using in Second Mind. Figure 74 shows the list of percepts available from the environment.

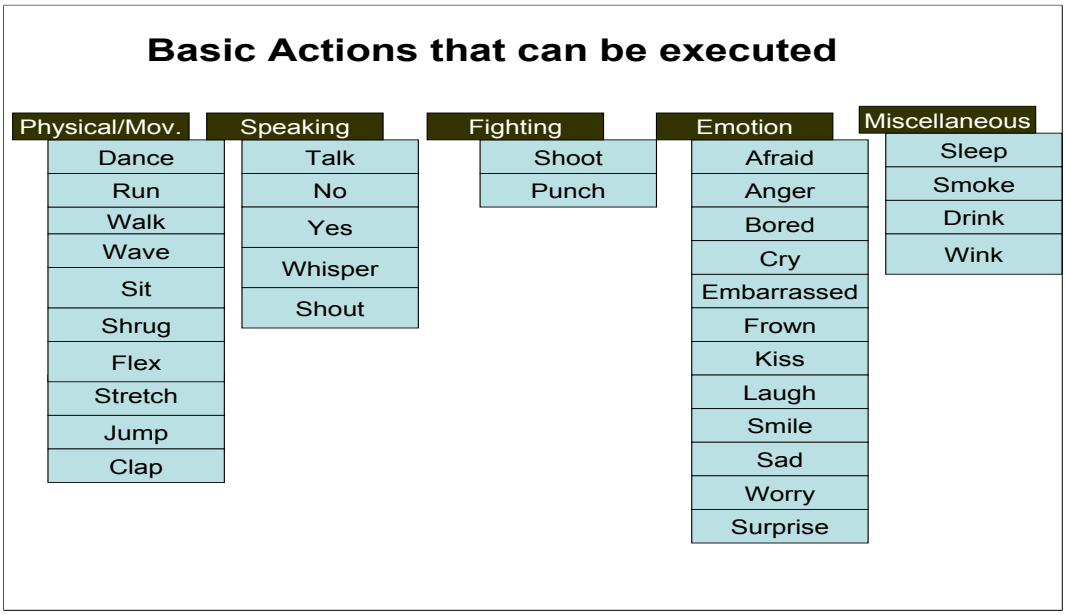



**Figure 73:** The figure shows the basic level actions from second life

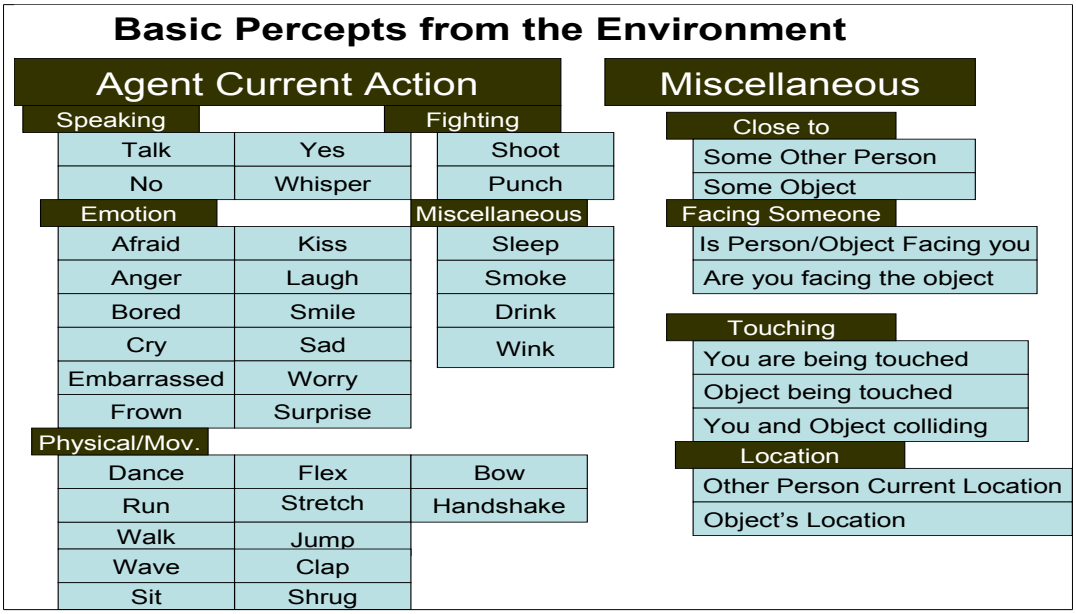



**Figure 74:** The figure shows the basic level percepts from second life

## *A.2 Interview Guide Paper Prototype*

1. **Were you able to understand the interfaces and the terminology used in the interfaces? If not, how would these be improved?.**

   _____

   _____

2. **Was it easy or difficult to use the system? How so? .**

   _____

   _____

3. **Were you able to achieve your given objective? If no, why not? .**

   _____

   _____

4. **Did you enjoy the tasks you were required to do ? If not, why not?.**

   _____

   _____

5. **Would you use this system again for doing these tasks, or would you like a different approach ? If different, please explain.**

   _____

   _____

6. **Did you find the interaction with the system enjoyable?. Which parts did you enjoy?**

   _____

   _____

7. **How would you like the authoring environment to be improved? What other features would you want in the development environment?**

   _____

   _____

8. **Other comments**

   _____

   _____

## *A.3 Quantitative Questionnaire Paper Prototype*

9. **How difficult was it to create the behaviors?**

   not difficult at all □—□—□—□—□ very difficult

10. **How difficult was it to create the advancing conditions?**

    not difficult at all □—□—□—□—□ very difficult

11. **How difficult was it to create failure condition?**

    not difficult at all □—□—□—□—□ very difficult

12. **How difficult was it to create success condition?**

    not difficult at all □—□—□—□—□ very difficult

13. **How difficult was it to create progress condition?**

    not difficult at all □—□—□—□—□ very difficult

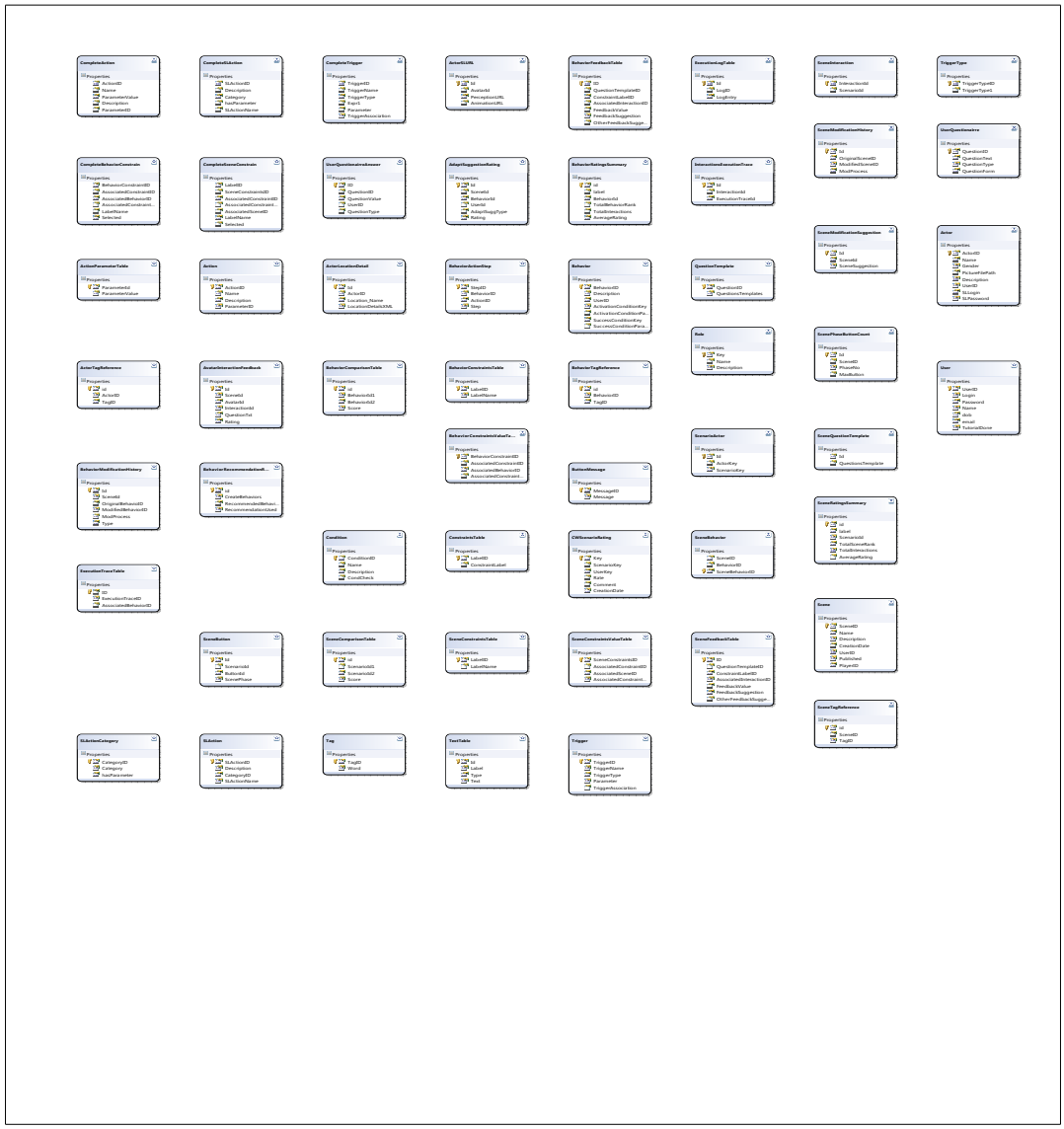


**Figure 75:** The figure shows the second mind database tables

## *A.4 Second Mind Database*

Figure 75 shows the Database tables used in Second Mind.

# APPENDIX B

# USER STUDY PHASE I

## *B.1 Mean, Standard Deviation and Effect Size*

Table 10 shows means, standard deviation and effect size across various measures for phase I. The categories that have been used for Table 10 are listed next.

**Table 10:** The table shows mean, standard deviation and effect size (Eta-squared) across various quantitative measure calculated for various author categories

| Cat. No. | AI beh. prog. | Prog. | SL Non Prog. | Non-prog. | Acting back-ground | UI de-signer | Effect Size |
|---|---|---|---|---|---|---|---|
| 1 | 4.30(0.64) | 3.10(0.46) | 4.40(0.66) | 4.57(0.59) | 4.70(0.46) | 4.30(0.46) | 0.49 |
| 2 | 4.00(0.77) | 3.90(0.36) | 3.90(0.54) | 4.07(0.70) | 4.10(0.30) | 3.80(0.62) | 0.21 |
| 3 | 3.80(0.75) | 3.00(0.63) | 4.00(1.10) | 3.86(0.99) | 4.40(0.66) | 3.00(0.40) | 0.28 |
| 4 | 1.10(0.20) | 1.00(0.20) | 1.10(0.23) | 1.14(0.10) | 1.00(0.20) | 1.20(0.25) | 0.02 |
| 5 | 2.25(0.34) | 1.90(0.43) | 2.38(0.20) | 2.21(0.37) | 3.00(0.01) | 1.85(0.23) | 0.62 |
| 6 | 2.25(0.25) | 1.54(0.16) | 2.15(0.23) | 2.01(0.06) | 3.00(0.01) | 1.75(0.25) | 0.86 |
| 7 | 7.80(0.40) | 8.50(0.47) | 8.00(0.01) | 8.00(0.01) | 8.60(0.49) | 7.90(0.30) | 0.49 |
| 8 | 1.23(0.10) | 2.16(0.20) | 2.47(0.32) | 2.77(0.23) | 2.69(0.32) | 2.16(0.19) | 0.83 |
| 9 | 2.40(0.29) | 1.96(0.35) | 2.46(0.30) | 2.30(0.28) | 2.91(0.09) | 2.10(0.17) | 0.57 |
| 10 | 13.00(3.41) | 14.60(2.29) | 13.40(1.80) | 14.93(1.45) | 15.20(2.20) | 15.20(1.60) | 0.23 |
| 11 | 0.75(0.19) | 1.38(0.23) | 1.26(0.54) | 1.00(0.11) | 1.12(0.44) | 1.56(0.30) | 0.38 |
| 12 | 1.03(0.08) | 0.79(0.20) | 1.02(0.04) | 1.04(0.10) | 1.19(0.13) | 0.93(0.17) | 0.39 |
| 13 | 55.20(4.12) | 67.99(4.22) | 71.08(6.21) | 71.84(3.82) | 84.67(4.45) | 68.50(2.96) | 0.80 |
| 14 | 0.95(0.14) | 1.02(0.22) | 1.03(0.13) | 1.06(0.12) | 0.90(0.06) | 0.90(0.04) | 0.22 |
| 15 | 11.50(2.58) | 8.50(1.11s) | 10.60(1.56) | 7.21(0.91) | 19.80(0.64) | 9.50(2.25) | 0.87 |

## *B.2 Categories*

Below are the categories that have been used for Table 10.

1. Were you able to author the scenes and behaviors you had in mind?

2. Rate your understanding of the interfaces and the terminology (like scenes, behaviors, constraints, behaviors, players etc) used in the system?

3. Rate your experience using the system?

4. Rate difficulty level of using the system?

5. Gestures per behaviors

6. Verbal Actions per behavior

7. Behaviors per scene

8. Trigger Creation Time per behavior
9. Action Creation Time per behavior
10. Time Spent in Second Life
11. Trigger Correction per behavior
12. Action correction per behavior
13. Total Authoring time
14. Time spent creating verbal action
15. Words per Verbal Talk action

## *B.3 Scheffé Post-hoc test results for Phase I*

The table 11 shows results from post-hoc analysis for phase I.

Table 11: The table shows post-hoc across various quantitative measure calculated for various author categories during phase I

| **$F_{Scheffe}$ Test Values where $F_{criticalvalue}$ = 16.75 for p=0.01** | | |
|---|---|---|
| **Gestures per behavior** | Acting vs AI | 30.82 |
| | UI vs AI | 8.77 |
| | Programmers vs AI | 6.71 |
| | AI vs Non Programmers | 0.07 |
| | Programmers vs non-programmers | 5.41 |
| | UI vs non-programmers | 7.27 |
| | Second Life vs AI | 0.93 |
| **Continued on next page** | | |

| Table 11 – continued from previous page | | |
|---|---|---|
| | Second Life NP vs non-programmers | 1.50 |
| | Second Life NP vs programmers | 12.62 |
| **Verbal action per behavior** | Acting vs AI | 79.40 |
| | UI vs AI | 35.29 |
| | Programmers vs AI | 71.15 |
| | AI vs Non Programmers | 7.84 |
| | Programmers vs non-programmers | 31.75 |
| | UI vs non-programmers | 9.86 |
| | Second Life vs AI | 1.41 |
| | Second Life vs non-programmers | 2.60 |
| | Programmer vs UI designer | 6.22 |
| **Able to author behaviors** | Acting vs AI | 2.55 |
| | UI vs AI | 0.00 |
| | Programmers vs AI | 22.92 |
| | AI vs Non Programmers | 1.17 |
| | Programmers vs non-programmers | 34.46 |
| | UI vs non-programmers | 1.17 |
| | Second Life vs AI | 0.16 |
| | Second Life vs non-programmers | 0.47 |
| **Behaviors per Scene** | Acting vs AI | 25.96 |
| | UI vs AI | 0.41 |
| | Programmers vs AI | 19.87 |
| | AI vs Non Programmers | 1.62 |
| | Programmers vs non-programmers | 10.14 |
| | UI vs non-programmers | 0.41 |
| **Continued on next page** | | |

| Table 11 – continued from previous page | | |
|---|---|---|
| | Second Life vs AI | 1.62 |
| | Second Life vs non-programmers | 0.00 |
| | Programmer vs Acting | 0.41 |
| **Trigger creation time** | Acting vs AI | 181.71 |
| | UI vs AI | 73.73 |
| | Programmers vs AI | 73.73 |
| | AI vs Non Programmers | 202.54 |
| | Programmers vs non-programmers | 31.87 |
| | UI vs non-programmers | 31.87 |
| | Second Life vs AI | 131.07 |
| | Second Life vs non-programmers | 7.75 |
| | Acting vs non-programmers | 0.57 |
| | Acting vs Second Life | 4.13 |
| | Second Life vs UI designer | 8.19 |
| | UI designer vs Programmer | 0.00 |
| | Acting vs UI designer | 0.00 |
| **Action creation time:** | Acting vs AI | 18.27 |
| | UI vs AI | 6.32 |
| | Programmers vs AI | 13.60 |
| | AI vs Non Programmers | 0.70 |
| | Programmers vs non-programmers | 8.12 |
| | UI vs non-programmers | 2.81 |
| | Second Life vs AI | 0.25 |
| | Second Life vs non-programmers | 1.80 |
| | Second Life vs UI designer | 9.10 |
| **Continued on next page** | | |

| Table 11 – continued from previous page | | |
|---|---|---|
| | Second Life vs Programmer | 15.56 |
| **Trigger Corr. time per behavior:** | Acting vs AI | 5.86 |
| | UI vs AI | 28.07 |
| | Programmers vs AI | 16.98 |
| | AI vs Non Programmers | 2.67 |
| | Programmers vs non-programmers | 6.18 |
| | UI vs non-programmers | 13.42 |
| | Second Life vs AI | 11.13 |
| | Second Life vs non-programmers | 2.89 |
| **Action Corr. time per behavior:** | Acting vs AI | 7.21 |
| | UI vs AI | 2.82 |
| | Programmers vs AI | 16.21 |
| | AI vs Non Programmers | 0.05 |
| | Programmers vs non-programmers | 18.00 |
| | UI vs non-programmers | 3.59 |
| | Second Life vs AI | 0.03 |
| | Second Life vs non-programmers | 0.15 |
| **Total Creation time:** | Acting vs AI | 216.24 |
| | UI vs AI | 44.04 |
| | Programmers vs AI | 40.73 |
| | AI vs Non Programmers | 68.97 |
| | Programmers vs non-programmers | 3.70 |
| | UI vs non-programmers | 2.78 |
| | Second Life vs AI | 62.79 |
| | Second Life vs non-programmers | 0.14 |
| **Continued on next page** | | |

| Table 11 – continued from previous page | | |
|---|---|---|
| | Acting vs non-programmers | 40.97 |

## *B.4 Analysis of Scheffé Post-hoc test*

- **Gesture per behavior:** Acting background non-programmer had the highest gesture per behavior. Authors in other categories had statistical similarity.
- **Verbal actions per behavior:** Acting background non-programmer had the highest verbal actions per behavior followed by Second Life non-programmer, AI behavior programmer and Non-programmer who had statistically similar verbal actions per behavior. UI designer and Programmer had statistically similar verbal actions per behavior and the lowest among all the author categories.
- **Were you able to author scenes and behaviors you had in mind?:** Programmer had the lowest rating. Authors in other categories had statistical similarity.
- **Behaviors per Scene:** Acting background non-programmer and Programmers had the highest statistically similar behaviors per scene. Authors in other categories had statistical similarity.
- **Trigger creation time:** Non-programmers, Second Life non-programmer and Acting background non-programmers had the highest statistically similar time. UI designer and the programmers had the next highest statistical similar. AI behavior programmers had the lowest time.
- **Action creation time:** Acting background non-programmer had the highest action creation time. Authors in other categories had statistical similarity.
- **Trigger correction time:** AI behavior programmer had the lowest trigger correction time though it is statistically similar to non-programmers, acting

background non-programmers and Second Life non-programmer. UI designer had statistical similar correction time to non-programmers, acting background non-programmers and Second Life non-programmer and programmer.

- **Action correction time:** Programmer had the lowest action correction time. Authors in other categories had statistical similarity.

- **Total Creation time:** Acting background non-programmer had the highest creation time and the AI behavior programmer the lowest. Authors in other categories had statistical similarity.

## *B.5 Quantitative Questionnaire Phase I study*

**14. Were you able to author the scenes and behaviors you had in mind? (choose only one)?**

- □ Not at all.
- □ Only a few.
- □ Sometimes yes, sometimes no.
- □ Yes, most of them.
- □ All of them.

**15. How would you rate the Second Mind interface in terms of allowing you to create the kind of behaviors you wanted to author?**

Wasn't able to create what I wanted □—□—□—□—□ Was able to create what I wanted

**16. Rate your understanding of the interfaces and the terminology (like scenes, behaviors, constraints, behaviors, players etc) used in the system?**

not able to understand □—□—□—□—□ understood completely

**17. Rate your experience using the system?**

did not enjoy it all □—□—□—□—□ enjoyed a lot using it

**18. Rate difficulty level of using the system?**

not difficult at all □—□—□—□—□ very difficult

## *B.6 Background and Pre-screening Questionnaire Phase I study*

**19. Gender (M/F):** _____

**20. Age**: _____

**21. Occupation**: _____

**22. Education**: _____

**23. Education**: _____

**24. Highest degree achieved (Ph.D., M.S:, B.S., etc.)?**: _____

**25a. Computer Gaming Experience on average (choose only one)?**

- ☐ None (I play at most 1 hour per month).
- ☐ Low (I play seldom, approx 1 hour per week)
- ☐ Medium (I play around 1 hour a day)
- ☐ High (I play more than 1 hour a day)
- ☐ Very high (I play more than 2 hours a day)

**26. How many hours per week do you spend playing video games?**: _____

**27. Game genres played (multiple choice is possible) ?**

- ☐ Action Adventure.
- ☐ Role-Playing
- ☐ First Person Shooters
- ☐ Turn-Based Strategy
- ☐ Real-Time Strategy
- ☐ Adventure
- ☐ Sports
- ☐ Puzzle

□ Multi-Player Online

□ Other (please specify) _____

**28. Computer Programming Experience (choose only one) ?**

□ None (I do not how to program)

□ Low (I am a beginner in programming)

□ Medium (I have some experience in languages through small project)

□ High (I can write rather large and complex programs)

□ Very high (I am a professional coder)

**29. Design Experience (choose only one)?**

□ None (I never design anything)

□ Low (I am designer by hobby)

□ Medium (I have some experience in designing of small project)

□ High (High (I often have to design for work)

□ Very high (I am a professional designer)

**30. Which kind of design field are you into (multiple choice is possible)?**

□ Graphic design

□ User interface design

□ Industrial design

□ Other (please specify) _____

**31. Story Creation Experience (choose only one) ?**

□ None

- ☐ Low (I wrote just 1-2 stories)
- ☐ Medium (I wrote a few stories and took some courses on that)
- ☐ High (I wrote some long stories with complex protagonists)
- ☐ Very High (I write stories as part of my main activity/job)

**32. Virtual World (VW) Experience (choose only one) ?**

- ☐ None
- ☐ Low (I spend about approx 1 hour/week in a VW)
- ☐ Medium (I spend about approx 7 hours/week in a VW)
- ☐ High (I spend about approx 2 hours/day in a VW)
- ☐ Very High (I spend over 2 hours/day in a VW)

**33. Video Editing tools Experience (choose only one) ?**

- ☐ None
- ☐ Low (I used such tools only a few times)
- ☐ Medium (I happen to use such tools now and then i.e. to create clips)
- ☐ High (I use such tools often)
- ☐ Very High (I use such tools on a daily basis e.g. as part of my activity/job)

**34. Second Life (SL) Experience (choose only one) ?**

- ☐ None
- ☐ Low (I was in only 1-2 times for curiosity)
- ☐ Medium (I spend about approx 7 hours/week in SL)
- ☐ High (I spend about approx 2 hours/day in SL)
- ☐ Very High (I spend over 2 hours/day in SL)

## *B.7 Interview Guide Phase I study*

35. **Were you able to author the scene and behavior you had in mind? If not, why not?.**

36. **Were there parts of the system that really worked (helping you create behaviors and scenes)? If yes, which ones and why?**

37. **Did the interface present any limitation on the kind of behaviors and scenes you had in mind? If yes, what kind?.**

38. **Would you use this system again for authoring behaviors and scenes, or would you like a different approach? If different, please explain.**

39. **Was it easy or difficult to use the system? Please explain?.**

40. **Did you find the interaction with the system enjoyable? Which parts did you enjoy most and why?.**

41. **Were you able to author the scene and behavior you had in mind? If not, why not?.**

    _____

    _____

42. **Were there parts you did not like? Why?.**

    _____

    _____

43. **How would you like the development environment to be improved? What other features would you want in the development environment?**

    _____

    _____

44. **Did the scene that was authored play back as expected? If not, why not?.**

    _____

    _____

45. **What is the strongest point of the system, if any?.**

    _____

    _____

46. **What is the weakest point of the system, if any?.**

    _____

    _____

47. **What is the main limitation of the system, if any?.**

    _____

    _____

**48. Other comments?**

_____

_____

# APPENDIX C

# USER STUDY PHASE II

## *C.1 Questionnaire for Phase II*

**49. Please rate your overall experience interacting with the avatar?**

didn't enjoy at all □—□—□—□—□ enjoyed a lot

**50. Please rate avatar's overall performance?**

didn't act well at all □—□—□—□—□ acted very well

**Table 12:** The table shows mean, standard deviation and effect size (eta-squared) across various quantitative measure calculated for various author categories during phase II

| Cat. No. | AI beh. prog. | Prog. | SL Non Prog. | Non-prog. | Acting back-ground | UI de-signer | Effect Size |
|---|---|---|---|---|---|---|---|
| 1 | 3.10(0.12) | 2.11(0.36) | 3.07(0.33) | 3.02(0.21) | 3.71(0.14) | 2.61(0.16) | 0.88 |
| 2 | 3.20(0.14) | 2.21(0.28) | 3.03(0.26) | 3.15(0.24) | 3.61(0.14) | 2.59(0.18) | 0.89 |

## *C.2 Mean, Standard Deviation and Effect Size for Phase II*

Table 12 shows mean, standard deviation and effect size across various quantitative measures for phase II.

## *C.3 Categories*

- Please rate your overall experience interacting with the avatar?
- Please rate avatar's overall performance?

## *C.4 Scheffé Post-hoc test results for Phase II*

Table 13 shows results of Scheffé post-hoc tests for Phase II.

Table 13: The table shows post-hoc across various quantitative measure calculated for various author categories during phase II

| **$F_{Scheffe}$ Test Values where $F_{criticalvalue}$ = 11.85 for p=0.01** | | |
|---|---|---|
| **Overall experience interacting** | AI vs Acting | 26.51 |
| | Acting vs Prog. | 189.26 |
| | Acting vs Non-programmer | 39.70 |
| | UI designer vs Programmer | 1.06 |
| | AI vs Programmer | 71.82 |
| | AI vs non-programmer | 0.51 |
| | AI vs Second Life | 0.04 |
| | Non-programmer vs Second Life | 0.27 |
| | UI designer vs Non-Programmer | 57.08 |
| **Avatar's overall performance** | AI vs Acting | 12.07 |
| | Acting vs Prog. | 145.01 |
| | Acting vs Non-programmer | 17.73 |
| | UI designer vs Programmer | 0.89 |
| | AI vs Programmer | 71.87 |
| | AI vs non-programmer | 0.19 |
| | AI vs Second Life | 2.05 |
| | Non-programmer vs Second Life | 1.29 |
| | UI designer vs Non-Programmer | 62.99 |

**Table 14:** The table shows pearson correlation across various quantitative measure calculated for various author categories during phase II

| Cat. No. | Prog | AI beh. prog. | UI designer | Non-prog. | Second Life Non Prog. | Acting background |
|---|---|---|---|---|---|---|
| 1 | 0.63 | 0.49 | 0.46 | 0.41 | 0.77 | -0.58 |
| 2 | 0.48 | -0.13 | -0.16 | -0.20 | -0.05 | -0.61 |
| 3 | 0.89 | -0.16 | -0.36 | 0.55 | 0.25 | 0.63 |
| 4 | -0.03 | 0.39 | 0.46 | 0.55 | 0.56 | 0.33 |
| 5 | -0.77 | 0.66 | -0.44 | 0.12 | 0.25 | -0.62 |
| 6 | 0.67 | -0.07 | 0.85 | 0.00 | -0.11 | -0.31 |
| 7 | -0.08 | 0.17 | 0.06 | 0.21 | 0.34 | -0.41 |
| 8 | 0.37 | 0.15 | -0.51 | 0.20 | 0.11 | 0.46 |
| 9 | -0.20 | 0.20 | -0.54 | 0.37 | 0.16 | -0.71 |
| 10 | 0.52 | 0.36 | 0.40 | 0.26 | 0.45 | -0.34 |
| 11 | -0.41 | 0.42 | 0.36 | -0.12 | -0.52 | 0.62 |
| 12 | 0.13 | -0.08 | -0.16 | 0.03 | 0.20 | -0.64 |

## *C.5 Pearson Correlation Analysis for Phase II*

Table 14 shows results from pearson correlation analysis for phase II. Categories for the table are shown later on.

## *C.6 Categories*

1. Gestures per behaviors
2. Verbal Actions per behaviors
3. Time taken to create a behavior
4. Behaviors per scene
5. Trigger Creation Time
6. Action Creation Time
7. Second Life Time
8. Trigger Correction Time
9. Action correction per behavior
10. Total Authoring Time
11. Time spent creating verbal action
12. Word per Verbal Talk action

**Table 15:** The table shows constraint scores assigned by the authors in relation to player feedback scores in phase II for various author categories

| Author Type. | Constraint Score | Player Feedback Score | Per. Difference |
|---|---|---|---|
| AI prog. | 4191.75 | 2916 | 43.75% |
| Acting back. | 4243.50 | 3382.5 | 25.45% |
| Second Life | 4140 | 2856 | 44.96% |
| Non-prog. | 6287.63 | 4355.78 | 44.35% |
| Prog. | 4295.25 | 2651.85 | 61.97% |
| UI des. | 4088.25 | 2512.20 | 62.74% |

## *C.7 Constraint Match Score*

Table 15 shows constraint scores assigned by the authors in relation to player feedback scores in phase II.

# APPENDIX D

# USER STUDY PHASE III

## *D.1 Quantitative Questionnaire and Interview Guide*

**51. How would you rate the suggestions where you had to modify the behaviors yourself (the first part of the evaluation)?**

not helpful at all □—□—□—□—□ very helpful

**52. Please explain your answers to the above question. Was the feedback helpful? Why or why not?.**

_____

_____

_____

_____

**53. How would you rate the suggestions where you were provided detailed suggestions on how to modify the behaviors by the system (the second part of the evaluation)?**

not helpful at all □—□—□—□—□ very helpful

**54. Please explain your answer to the above. Was the feedback helpful? Why or why not? How would you compare it against the feedback you received as numerical rating)?.**

_____

_____

_____

_____

55. **Would it be useful if you were able to get more detailed suggestion than the ones that were currently presented to you in the interface? If yes, please explain?.**

_____

_____

_____

_____

56. **How could the detailed suggestions (related to part 2) that are given be improved? What more can be added to them, if anything?.**

_____

_____

_____

_____

**Table 16:** The table shows mean, standard deviation and effect size (eta-squared) across various quantitative measure calculated for various author categories during phase III

| Cat. No. | AI beh. prog. | Second Life Non Prog. | Acting background | Effect Size |
|---|---|---|---|---|
| 1 | 3.30(1.24) | 3.30(0.70) | 3.10(0.87) | 0.66 |
| 2 | 4.20(1.50) | 4.50(0.48) | 4.30(0.54) | 0.77 |
| 3 | 3.20(0.42) | 3.30(0.54) | 1.70(0.38) | 0.76 |
| 4 | 2.10(0.59) | 2.50(0.51) | 1.80(0.39) | 0.89 |
| 5 | 3.10(0.41) | 2.90(0.64) | 1.90(0.30) | 0.74 |
| 6 | 4.10(0.35) | 4.20(0.35) | 4.50(0.44) | 0.17 |
| 7 | 3.75(0.46) | 3.60(0.33) | 2.80(0.53) | 0.95 |
| 8 | 1.10(0.50) | 2.00(0.52) | 1.90(0.32) | 0.56 |
| 9 | 4.30(0.39) | 4.00(0.42) | 4.40(0.25) | 0.96 |
| 10 | 3.30(1.28) | 3.60(0.63) | 4.50(2.65) | 0.58 |
| 11 | 8.00(1.09) | 8.10(1.11) | 11.60(1.44) | 0.00 |
| 12 | 1.00(0.01) | 0.1(0.01) | 1.00(0.01) | 0.73 |
| 13 | 1.10(0.13) | 2.00(0.24) | 1.60(0.24) | 0.57 |
| 14 | 2.80(0.16) | 2.00(0.21) | 1.40(0.24) | 0.06 |
| 15 | 0.30(0.41) | 0.90(0.26) | 0.50(0.25) | 1.00 |
| 16 | 0.00(0.00) | 0.00(0.00) | 0.00(0.00) | 1.00 |

## *D.2 Mean, Standard Deviation and Effect Size*

Table 16 shows mean, standard deviation and effect size for various quantitative measures for phase III. Categories used in the table are shown later on.

## *D.3 Category*

1. How would you rate the suggestions where you had to modify the behaviors yourself (the first part of the evaluation)?
2. How would you rate the suggestions where you were provided detailed suggestions on how to modify the behaviors by the system (the second part of the evaluation)?
3. New Behaviors Added
4. Modified Existing Behavior
5. Addition Suggestions
6. Ratings Addition Suggestion
7. Ratings Existing Behavior
8. Trigger Suggestion
9. Rating Trigger Suggestion
10. (Author) Behavior Actions Deleted in Step 1
11. (Author) Behavior Actions Added in Step 1
12. (Author) New Behavior Added in Step 1
13. Trigger Suggestion Accepted (Verbatim) in Step 2
14. Addition Suggestion Accepted (Verbatim) in Step 2
15. Accepted Suggestion with changes in Step 2
16. Accepted Trigger Suggestion with changes in Step 2

## *D.4 Scheffé Post-hoc test results for Phase III*

Table 17 shows Scheffé post-hoc tests results for Phase III.

Table 17: The table shows post-hoc across various quantitative measure calculated for various author categories during phase III

| **$F_{Scheffe}$ Test Values where $F_{criticalvalue}$ = 10.98 for p=0.01** | | |
|---|---|---|
| **Trigger Suggestion** | AI vs Second Life | 20.08 |
| | AI vs Acting | 15.87 |
| | Acting vs Second Life | 0.25 |
| **Ratings Addition Suggestion** | AI vs Second Life | 0.10 |
| | AI vs Acting | 1.64 |
| | Acting vs Second Life | 0.92 |
| **New Behavior Added** | AI vs Second Life | 0.16 |
| | AI vs Acting | 35.12 |
| | Acting vs Second Life | 39.96 |
| **Modified Existing Behavior** | AI vs Second Life | 3.13 |
| | AI vs Acting | 1.76 |
| | Acting vs Second Life | 9.59 |
| **Addition Suggestion** | AI vs Second Life | 0.65 |
| | AI vs Acting | 23.28 |
| | Acting vs Second Life | 16.17 |
| **Ratings Existing Behavior** | AI vs Second Life | 0.29 |
| | AI vs Acting | 11.50 |
| | Acting vs Second Life | 8.16 |
| | AI vs Second Life | 0.93 |
| **Rating Trigger Suggestions** | Continued on next page | |

| Table 17 – continued from previous page | | |
|---|---|---|
| | AI vs Acting | 0.10 |
| | Acting vs Second Life | 1.65 |

## *D.5 Analysis of Scheffé Post-hoc test*

- **Trigger Suggestion:** AI behavior programmer got the lowest trigger suggestions. Authors in other categories had statistically similarity.
- **Ratings Addition Suggestion:** Ratings across all author category had statistically similarity.
- **New Behavior Added:** Acting background non-programmer had the lowest new behavior additions. Authors in other categories had statistically similarity.
- **Modified Existing Behavior:** Ratings across all author category had statistically similarity.
- **Addition Suggestion:** Acting background non-programmer had the lowest number of addition suggestions. Authors in other categories had statistically similarity.
- **Ratings Existing Behavior:** Acting background non-programmer and AI behavior programmer had statistical similarity.
- **Rating Trigger Suggestions:** Ratings across all author category had statistically similarity.

# APPENDIX E

# USER STUDY PHASE IV

## *E.1 Mean, Standard Deviation and Effect Size for Phase IV*

Table 18 shows mean, standard deviation and effect size for various quantitative measures for phase IV.

**Table 18:** The table shows mean, standard deviation and effect size (eta-squared) across various quantitative measure calculated for various author categories during phase IV where original=behaviors from phase I, Author=behaviors adapted by the authors by looking at the numerical ratings, AI=behaviors adapted by the AI repair system and Mixed=behaviors adapted in combination with AI and authors. 1, 2 are categories given later on.

| Cat. No. | AI beh. prog. | Second Life Non Prog. | Acting background | Effect Size |
|---|---|---|---|---|
| Original(1) | 3.08(0.18) | 3.02(0.38) | 3.56(0.28) | 0.47 |
| Author(1) | 3.40(0.27) | 3.34(0.21) | 3.88(0.18) | 0.71 |
| AI(1) | 3.72(0.27) | 3.72(0.16) | 4.12(0.22) | 0.61 |
| Mixed(1) | 4.18(0.11) | 4.06(0.13) | 4.56(0.15) | 0.60 |
| Original(2) | 3.02(0.11) | 2.960(0.20) | 3.56(0.23) | 0.51 |
| Author(2) | 3.5(0.16) | 3.18(0.12) | 3.88(0.41) | 0.79 |
| AI(2) | 3.82(0.12) | 3.60(0.01) | 4.15(0.21) | 0.80 |
| Mixed(2) | 4.14(0.16) | 4.16(0.08) | 4.66(0.09) | 0.86 |

## *E.2 Constraint Match Score*

Table 19 shows constraint scores assigned by the authors in relation to player feedback scores in phase IV.

| Author Type. | Constraint Score | Player Feedback Score | Per. difference |
|---|---|---|---|
| AI prog.(Orig.) | 3656.25 | 2587.5 | 41.30% |
| Acting back.(Orig. | 4243.50 | 3382.5 | 25.45% |
| Second Life(Orig.) | 3881.25 | 2756.25 | 40.82% |
| AI prog.(Self Mod.) | 4488 | 3432 | 30.77% |
| Acting back.(Self Mod.) | 5016 | 4026 | 24.59% |
| Second Life(Self Mod.) | 4686 | 3564 | 31.48% |
| AI prog.(AI Mod.) | 6171 | 5626.5 | 9.68% |
| Acting back.(AI Mod.) | 6987.75 | 6261.75 | 11.59% |
| Second Life(AI Mod.) | 6352.5 | 5717.25 | 11.11% |
| AI prog.(Mixed Mod.) | 8835 | 8021.25 | 10.14% |
| Acting back.(Mixed Mod.) | 8951.25 | 8253.75 | 8.45% |
| Second Life(Mixed Mod.) | 9021 | 8137.5 | 10.86% |

**Table 19:** The table shows constraint scores assigned by the authors in relation to player feedback scores in phase IV for various author categories

## *E.3 Category*

1. Please rate your overall experience interacting with the avatar (Original Behaviors)?

2. Please rate avatar's overall performance?

## *E.4 Scheffé Post-hoc test results for Phase IV*

Table 20 shows results from Scheffé post-hoc tests for phase IV.

Table 20: The table shows post-hoc across various quantitative measure calculated for various author categories during phase IV

| $\mathbf{F}_{Scheffe}$ **Test Values where** $\mathbf{F}_{criticalvalue}$ **= 8.61 for p=0.01** | | |
|---|---|---|
| **AI behavior programmer Cat=1** | Author modified vs AI modified | 10.47 |
| | AI modified vs Mixed modified | 21.63 |
| | Original vs Author modified | 10.47 |
| **AI behavior programmer Cat=2** | Author modified vs AI modified | 10.47 |
| | AI modified vs Mixed modified | 10.47 |
| | Original vs Author modified | 23.55 |
| **Acting non-programmer Cat=1** | Author modified vs AI modified | 10.47 |
| | AI modified vs Mixed modified | 19.79 |
| | Original vs Author modified | 10.47 |
| **Acting non-programmer Cat=2** | Author modified vs AI modified | 10.47 |
| | AI modified vs Mixed modified | 26.58 |
| | Original vs Author modified | 10.47 |
| **SL non-programmer Cat=1** | Author modified vs AI modified | 14.76 |
| | AI modified vs Mixed modified | 11.81 |
| | Original vs Author modified | 10.47 |
| **SL non-programmer Cat=2** | Author modified vs AI modified | 18.03 |
| | AI modified vs Mixed modified | 32.05 |
| | Original vs Author modified | 4.95 |

# REFERENCES


[1] "Second life: http://www.secondlife.com/," 1 May 2008.

[2] "Second life animation list: http://wiki.secondlife.com/wiki/internal_animations," 27 June 2008.

[3] "Second life perception list: http://wiki.secondlife.com/wiki/category:lsl_detected," 28 June 2008.

[4] 3, S., "http://thesims3.ea.com/," 10 May 2008.

[5] Anderson, C., *The Long Tail: Why the Future of Business is Selling Less of More*. 2006.

[6] Anderson, J. R. and Jeffries, R., "Novice lisp errors: undetected losses of information from working memory," *Human Computer Interaction*, vol. 1, no. 2, pp. 107–131, 1985.

[7] Anderson, M. L. and Oates, T., "A review of recent research in metareasoning and metalearning," *AI Magazine*, vol. 28, pp. 7–16, 2007.

[8] Applied, G. M., Green, T. R. G., and Petre, M., "Usability analysis of visual programming environments: a 'cognitive dimensions' framework," *Journal of Visual Languages and Computing*, vol. 7, pp. 131–174, 1996.

[9] Arts, E., "http://www.ea.com/official/battlefield/battlefield2/us/," 2006.

[10] Bailey, B. P., "Demais: A behavior-sketching tool for early multimedia design," 2002.

[11] Barzilay, R., Mccullough, D., Rambow, O., Decristofaro, J., Korelsky, T., Lavoie, B., and Inc, C., "A new approach to expert system explanations," in *In 9thInternational Workshop on Natural Language Generation*, pp. 78–87, 1998.

[12] Beetz, M., "Structured reactive controllers a computational model of everyday activity," in *Proceedings of the Third International Conference on Autonomous Agents*, 2000.

[13] Bonasso, P., Firby, J., Gat, E., Kortenkamp, D., Miller, D., and Slack, M., "Experiences with an architecture for intelligent reactive agents," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 9, pp. 237–256, 1997.

[14] Breazeal, C., Berlin, M., Brooks, A., Gray, J., and Thomaz, A. L., "Using perspective taking to learn from ambiguous demonstrations," 2006.

[15] Brooks, K. M., "Do story agents use rocking chairs? the theory and implementation of one model for computational narrative," in *Proceedings of the fourth ACM international conference on Multimedia*, MULTIMEDIA '96, (New York, NY, USA), pp. 317–328, ACM, 1996.

[16] Bruns, A., *Gatewatching: Collaborative Online News Production (Digital Formations)*. Peter Lang Publishing.

[17] Burea, I. A., "User-generated content social media and advertising: An overview," 2008.

[18] Buro, M., "Rts games as test-bed for real-time ai researche," in *Proceedings of the 7th Joint Conference on Information Science (JCIS 2003)*, pp. 481–484, 2003.

[19] Carroll, J., "Completing design in use: closing the appropriation cycle," in *ECIS*, 2004.

[20] Cazenave, T., "Metarules to improve tactical go knowledge," *Inf. Sci. Inf. Comput. Sci.*, vol. 154, no. 3-4, pp. 173–188, 2003.

[21] Ciborra, C., *The Labyrinths of Information: Challenging the Wisdom of Systems*. Oxford University Press Oxford, 2002.

[22] Content, W., Agamanolis, S., and Michael, V., "Feature article viper: A framework for responsive television."

[23] Conway, M. J., *Alice: Easy-to-Learn 3D Scripting for Novices*. PhD thesis, University of Virginia, 1997.

[24] Corp., B., "http://www.bioware.com/games/mdk2/," 2006.

[25] Corradini, A., Mehta, M., Bernsen, N.-O., and Charfuelan, M., "Animating an interactive conversational character for an educational game system," in *IUI '05: Proceedings of the 10th international conference on Intelligent user interfaces*, (New York, NY, USA), pp. 183–190, ACM, 2005.

[26] Corradini, A., Mehta, M., and Charfuelan, M., "Interacting with an animated conversational agent in a 3d graphical setting," in *Proceedings of the Workshop on Multimodal Interaction for the Visualization and Exploration of scientific data at the International Conference on Multimodal Interfaces (ICMI05)*, pp. 63–70, 2005.

[27] Cox, M. T., "Metacognition in computation: a selected research review," *Artif. Intell.*, vol. 169, no. 2, pp. 104–141, 2005.

[28] Cox, M. T. and Ram, A., "Introspective multistrategy learning: On the construction of learning strategies," tech. rep., 1996.

[29] Crawford, C., *Chris Crawford on Interactive Storytelling (New Riders Games)*. New Riders Games, 2004.

[30] Cypher, A., Halbert, D. C., Kurlander, D., Lieberman, H., Maulsby, D., Myers, B. A., and Turransky, A., eds., *Watch what I do: programming by demonstration*. Cambridge, MA, USA: MIT Press, 1993.

[31] Davies, S. P., "Display-based problelm solving stragegies in computer programming," in *Empirical Studies of Programmers: Sixth Workshop*. (Gray, W. D. and Boehm-Davis, D. A., eds.), pp. 59–76, Springer-Verlag, 1996.

[32] Dix, A., "Designing for appropriation," in *Proceedings of HCI 2007. The 21st British HCI Group Annual Conference*, 2007.

[33] El-Nasr, M. and Smith, B., "Learning through game modding," *ACM Computers in Entertainment*, vol. 4, no. 1, 2006.

[34] Entertainment, R., "http://www.relic.com/games/homeworld-2/," 2006.

[35] Firby, R. J., "An investigation into reactive planning in complex domains," in *AAAI-87*, pp. 202–206, 1987.

[36] Floyd, M. W., Esfandiari, B., and Lam, K., "A case-based reasoning approach to imitating robocup players," in *FLAIRS Conference*, pp. 251–256, 2008.

[37] Games, F., "http://www.firaxis.com/games/game_detail.php?gameid=6," 2006.

[38] Games, T., "http://www.vampirebloodlines.com/," 2006.

[39] Gat, E., "Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots," in *Proceedings of the AAAI Conference*, 1992.

[40] Georgeff, M. P. and Lansky, A. L., "Reactive reasoning and planning," in *AAAI*, pp. 677–682, 1987.

[41] Gillmore, D., *We the Media*. O Reilly, August 2004.

[42] Gilmore, D. J. and Smith, H. T., "An investigation of the utility of flowcharts during computer program debugging," *Int. J. Man-Mach. Stud.*, vol. 20, no. 4, pp. 357–372, 1984.

[43] Gomboc, D., Solomon, S., Core, M. G., Lane, H. C., and Lent, M. V., "Design recommendations to support automated explanation and tutoring," in *In Proceedings of the 14th Conference on Behavior Representation in Modeling and Simulation (BRIMS05), Universal*, 2005.

[44] Good, J. and Robertson, J., "Learning and motivational affordances in narrative-based game authoring," in *In Proceedings of 4th International Conference for Narrative and Interactive Learning Environments (NILE)*, 2005.

[45] Green, T. R. G., Bellamy, R. K. E., and Parker, M., "Comprehensibility of visual and textual programs: A test of superlativisms against the match-mismatch conjecture," in *Empirical Studies of Programmers:Sixth Workshop*, pp. 121–146, 1991.

[46] Green, T. and Petre, M., "When visual programs are harder to read than textual programs," in *In*, pp. 167–180, 1992.

[47] Heckel, P., *The Elements of Friendly Software Design*. Alameda, CA, USA: SYBEX Inc., 1994.

[48] Hernndez, A. M. and Resnick, M., "Empowering kids to create and share programmable media," in *Interactions*, 2008.

[49] Huebner, R., "Postmortem: Nihilistic softwares vampire: The masquerade redemption.," July 2000.

[50] Jenkins, H., Clinton, K., Purushotma, R., Robinson, J. A., and Weigel, M., "Confronting the challenges of participatory culture: Media education for the 21st century," tech. rep., 2006.

[51] Jeppesen, L. B., "The implications of "user toolkits for innovation"," IVS/CBS Working Papers 2002.

[52] Jeppesen, L. B. and Molin, M. J., "Consumers as co-developers learning and innovation outside the firm," Tech. Rep. 2003-01.

[53] Jhala, A., Bares, W., and Young, M., "Towards an intelligent storyboarding tool for 3d games," in *In Proceedings of Advances in Computer Entertainment Technology*, 2005.

[54] Kaelbling, L. P. and Rosenschein, S. J., "Action and planning in embedded agents," *Robotics and Autonomous Systems*, vol. 6, pp. 35–48, 1990.

[55] Kearns, M. and Singh, S., "Near-optimal reinforcement learning in polynomial time," *Machine Learning*, vol. 49, no. 2-3, pp. 209–232, 2002.

[56] Kelleher, C., *Motivating programming: using storytelling to make computer programming attractive to middle school girls*. PhD thesis, Pittsburgh, PA, USA, 2006. AAI3248491.

[57] Kelleher, C. and Pausch, R., "Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers," *ACM Computer Survey*, vol. 37, no. 2, pp. 83–137, 2005.

[58] Kriegel, M. and Aylett, R., "An authoring tool for an emergent narrative storytelling system."

[59] Kriegel, M. and Aylett, R., "Crowd-sourced ai authoring with enigma," in *Proceedings of the Third joint conference on Interactive digital storytelling*, ICIDS'10, (Berlin, Heidelberg), pp. 275–278, Springer-Verlag, 2010.

[60] Laird, J. E. and Rosenbloom, P. S., "The evolution of the soar cognitive architecture," in *In*, pp. 1–50, 1994.

[61] Lane, H. C., Core, M. G., van Lent, M., Solomon, S., and Gomboc, D., "Explainable artificial intelligence for training and tutoring," in *Proceeding of the 2005 conference on Artificial Intelligence in Education: Supporting Learning through Intelligent and Socially Informed Technology*, (Amsterdam, The Netherlands, The Netherlands), pp. 762–764, IOS Press, 2005.

[62] Language, L. T. P., "http://www.lua.org," 2008.

[63] Language, P. T. P., "http://www.python.org," 2008.

[64] Lastowka, G., "User-generated content and virtual worlds," *Journal of Entertainment and Technology Law*, vol. 893, 2008.

[65] Lebowitz, M., "Creating a story-telling universe," in *Proceedings of the Eighth international joint conference on Artificial intelligence - Volume 1*, (San Francisco, CA, USA), pp. 63–65, Morgan Kaufmann Publishers Inc., 1983.

[66] Lewis, C. and Olson, G., "Can principles of cognition lower the barriers to programming?," pp. 248–263, 1987.

[67] Mateas, M. and Stern, A., "A behavior language for story-based believable agents," *IEEE intelligent systems and their applications*, vol. 17, no. 4, pp. 39–47, 2002.

[68] Mateas, M. and Stern, A., "Facade: An experiment in building a fully-realized interactive drama.," in *Game Developer's Conference: Game Design Track*, 2003.

[69] Mateas, M., *Interactive drama, art and artificial intelligence*. PhD thesis, Pittsburgh, PA, USA, 2002.

[70] McDermott, D., "Transformational planing of reactive behavior.," tech. rep., Yale University, 1992. Research Report YALEU/DCS/RR-941.

[71] McNaughton, M., Cutumisu, M., Szafron, D., Schaeffer, J., Redford, J., and Parker, D., "Scriptease: Generative design patterns for computer role-playing games," in *19th IEEE International Conference on Automated Software Engineering*, pp. 88–99, 2004.

[72] Medler, B. and Magerko, B., "Scribe: A general tool for authoring interactive drama.," in *3rd International Conference on Technologies for Interactive Digital Storytelling and Entertainment*, pp. 139–150, 2006.

[73] Mehta, M., Onta nón, S., and Ram, A., "Using meta-reasoning to improve the performance of case-based planning," in *ICCBR '09: Proceedings of the 8th International Conference on Case-Based Reasoning*, (Berlin, Heidelberg), pp. 210–224, Springer-Verlag, 2009.

[74] Mehta, M., Ontanon, S., and Ram, A., *Adaptive Computer Games: Easing the Authorial Burden.* Charles River Media, 2008.

[75] Millington, I., *Artificial Intelligence for Games.* Morgan Kaufmann, 2006.

[76] Moshirnia, A., "Knowledge sharing structures within modification culture," in *Proceedings of IASTED: KSCE*, 2006.

[77] Nareyek, A., "Intelligent agents for computer games," in *Computers and Games, Second International Conference, CG 2000*, pp. 414–422, 2002.

[78] Nicolescu, M. N., *A framework for learning from demonstration, generalization and practice in human-robot domains.* PhD thesis, Los Angeles, CA, USA, 2003. Adviser-Maja J. Mataric.

[79] Nieborg, D. B., "Am i mod or not? - an analysis of first person shooter modification culture," 2005.

[80] Ohlen, J., Zeschuk, G., and Muzyka, R., "Postmortem: Biowares baldurs gate ii," March 2001.

[81] Ontanon, S., Mishra, K., Sugandh, N., and Ram, A., "On-line case-based planning," *Computational Intelligence*, vol. 26, no. 1, pp. 84–119, 2010.

[82] Orkin, J. and Roy, D., "Automatic learning and generation of social behavior from collective human gameplay," in *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '09, (Richland, SC), pp. 385–392, International Foundation for Autonomous Agents and Multiagent Systems, 2009.

[83] Pacey, A., *The culture of technology / Arnold Pacey.* B. Blackwell, Oxford, England :, 1983.

[84] Pane, J. F. and Myers, B. A., "Usability issues in the design of novice programming systems," tech. rep., Carnegie Mellon University, August 1996.

[85] Pearce, C., "Emergent authorship: the next interactive revolution," *Computers & Graphics*, vol. 26, no. 1, pp. 21–29, 2002.

[86] Peot, M. and Smith, D., "Conditional nonlinear planning," in *First International Conference on AI Planning Systems*, 1992.

[87] Picard, R. W., *Affective computing*. Cambridge, MA, USA: MIT Press, 1997.

[88] Prada, R., Paiva, A., Machado, I., and Gouveia, C., "you cannot use my broom! im the witch, youre the prince: Collaboration in a virtual dramatic game," in *Intelligent Tutoring Systems* (Cerri, S., Gouardres, G., and Paraguau, F., eds.), vol. 2363 of *Lecture Notes in Computer Science*, pp. 913–922, Springer Berlin / Heidelberg, 2002.

[89] Prgl, R. and Schreier, M., "Learning from leading-edge customers at the sims opening up the innovation process using toolkits," *Journal of R & D Management*, vol. 36, no. 3, pp. 237–250, 2006.

[90] Puga, G. F., Gómez-Martín, M. A., Díaz-Agudo, B., and González-Calero, P. A., "Dynamic expansion of behaviour trees," in *AIIDE*, 2008.

[91] Rabin, S., *AI Game Programming Wisdom II*. Charles River Media, 2004.

[92] Rao, R. P. N., Shon, A. P., and Meltzoff, A. N., "A bayesian model of imitation in infants and robots," in *In Imitation and Social Learning in Robots, Humans, and Animals*, Cambridge University Press, 2004.

[93] Resnick, M., Myers, B., Pausch, R., Nakakoji, K., Shneiderman, B., and Eisenberg, T. S. M., "Design principles for tools to support creative thinking, workshop on creativity support tools. pp. 25-36.," in *Design Principles for Tools to Support Creative Thinking, Workshop on Creativity Support Tools*, pp. 25–36, 2007.

[94] Saltzman, M., *Game Design: Secrets of the Sages*. Brady Publishing, 1999.

[95] Sauer, S., Osswald, K., Wielemans, X., and Stifter, M., "U-create: Creative authoring tools for edutainment applications," in *TIDSE 2006* (S. Göbel, R. Malkewitz, I. I., ed.), (Berlin Heidelberg), pp. 163–168, ZGDV, Springer-Verlag, Dec. 2006.

[96] Schwab, B., *AI Game Engine Programming*. Charles River Media, 2004.

[97] Shortliffe, E. H., *Computer-based medical consultations, MYCIN / Edward Hance Shortliffe*. Elsevier, New York :, 1976.

[98] Simpkins, C., Bhat, S., Isbell, Jr., C., and Mateas, M., "Towards adaptive programming: integrating reinforcement learning into a programming language," *SIGPLAN Not.*, vol. 43, pp. 603–614, October 2008.

[99] Skorupski, J., "Storyboard authoring of plan-based interactive dramas," in *Proceedings of the 4th International Conference on Foundations of Digital Games*, FDG '09, (New York, NY, USA), pp. 349–351, ACM, 2009.

[100] Skorupski, J., Jayapalan, L., Marquez, S., and Mateas, M., "Wide ruled: a friendly interface to author-goal based story generation," in *Proceedings of the 4th international conference on Virtual storytelling: using virtual reality technologies for storytelling*, ICVS'07, (Berlin, Heidelberg), pp. 26–37, Springer-Verlag, 2007.

[101] Spector, W., "Postmortem: Ion storms deus ex," November 2000.

[102] Spronck, P., Ponsen, M., Sprinkhuizen-Kuyper, I., and Postma, E., "Adaptive game ai with dynamic scripting," *Mach. Learn.*, vol. 63, no. 3, pp. 217–248, 2006.

[103] Spronck, P., Ponsen, M., Sprinkhuizen-Kuyper, I., and Postma, E., "Adaptive game ai with dynamic scripting," *Mach. Learn.*, vol. 63, pp. 217–248, June 2006.

[104] Strauss, A. and Corbin, J., *Basics of Qualitative Research: Grounded Theory Procedures and Techniques*. Sage, 1990.

[105] Stroulia, E. and Goel, A. K., "Functional representation and reasoning in reflective systems," *Journal of Applied Intelligence*, vol. 9, pp. 101–124, 1995.

[106] Toffler, A., *The third wave*. 1981.

[107] Tozour, P., *The Perils of AI Scripting*, pp. 541–547. Charles River Media Inc, 2002.

[108] Ubisoft, "farcry.ubi.com/," 2006.

[109] Ulam, P., Jones, J., and Goel, A. K., "Combining model-based meta-reasoning and reinforcement learning for adapting game-playing agents.," in *AIIDE*, 2008.

[110] Virmani, S., Kanetkar, Y., Mehta, M., Ontañón, S., and Ram, A., "An intelligent ide for behavior authoring in real-time strategy games," in *AIIDE*, 2008.

[111] von Hippel, E., *Democratizing Innovation*. The MIT Press, April 2005.

[112] Weld, D. S., Anderson, C. R., and Smith, D., "Extending graphplan to handle uncertainty and sensing actions," in *Proceedings of AAAI*, 1998.

[113] Yu, K., Wang, H., Liu, C., and Niu, J., "Interactive storyboard: Animated story creation on touch interfaces," in *Proceedings of the 5th International Conference on Active Media Technology*, AMT '09, (Berlin, Heidelberg), pp. 93–103, Springer-Verlag, 2009.

[114] Zagalo, N., Gbel, S., Torres, A., and Malkewitz, R., "Inscape: Emotion expression and experience in an authoring environment."

[115] Zang, P., Mehta, M., Mateas, M., and Ram, A., "Towards runtime behavior adaptation for embodied characters," in *IJCAI'2007*, pp. 1557–1562, 2007.